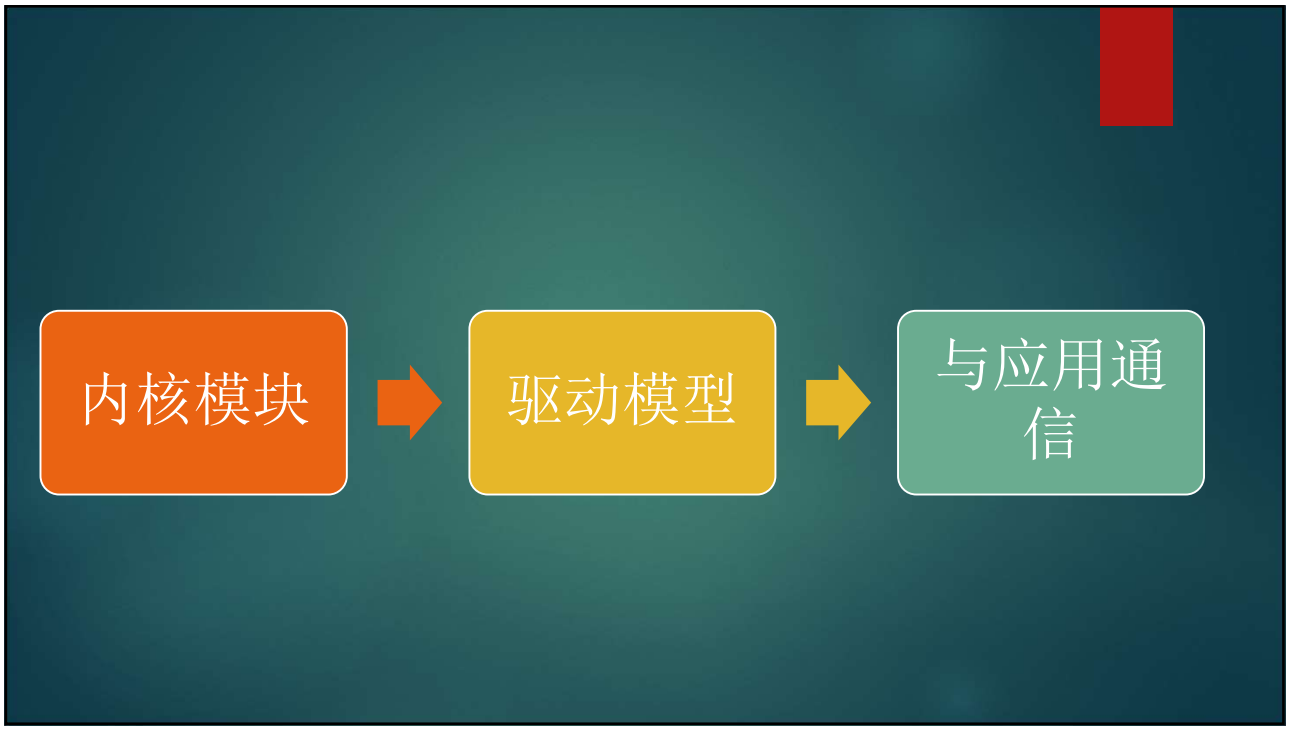
LINUX系统高级调试和优化

-- 内核模块

张银奎

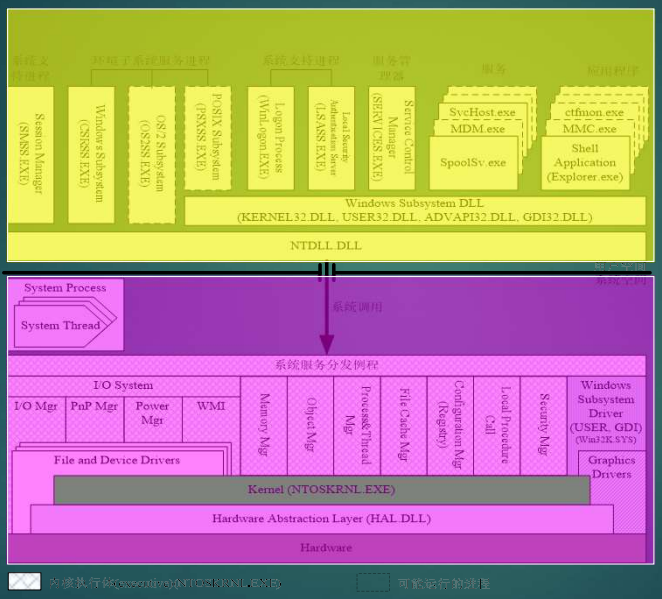
2016/8/18

1



2

乙儀攸乡



日

阳 ☰ 乾 天

天行健，君子自强不息

地势坤，君子厚德载物

阴 ☷ 坤 地

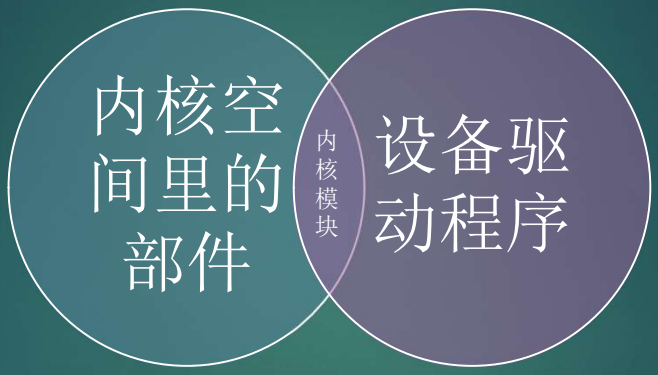
月

3

```
#0  udp_sendmsg (iocb=0xdb805e28, sk=0xede0b400, msg=0xdb805e78, len=35)
    at net/ipv4/udp.c:827
#1  0xc15fa360 in inet_sendmsg (iocb=0xdb805e28, sock=<optimized out>,
    msg=<optimized out>, size=35) at net/ipv4/af_inet.c:770
#2  0xc158300b in __sock_sendmsg_nosec (size=35, msg=0xdb805e78,
    sock=0xcf40f600, iocb=0xdb805e28) at net/socket.c:631
#3  __sock_sendmsg (size=35, msg=0xdb805e78, sock=0xcf40f600, iocb=0xdb805e28)
    at net/socket.c:639
#4  __sock_sendmsg (size=35, msg=0xdb805e78, sock=0xcf40f600, iocb=0xdb805e28)
    at net/socket.c:3485
#5  sock_sendmsg (sock=0xcf40f600, msg=0xdb805e78, size=35) at net/socket.c:650
#6  0xc1583792 in SYSC_sendto (fd=<optimized out>, buff=0xa18fb930, len=35,
    flags=16448, addr=0x0, addr_len=0) at net/socket.c:1796
#7  0xc1583dad in SyS_sendto (fd=50, buff=-1584416464, len=35, flags=16384,
    addr=0, addr_len=0) at net/socket.c:1763
#8  0xc1583deb in SYSC_send (flags=<optimized out>, len=<optimized out>,
    buff=<optimized out>, fd=<optimized out>) at net/socket.c:1811
#9  SyS_send (fd=50, buff=-1584416464, len=35, flags=16384)
    at net/socket.c:1809
#10 0xc1584864 in SYSC_socketcall (args=0xa18fb7f0, call=9)
    at net/socket.c:2516
#11 SyS_socketcall (call=9, args=-1584416784) at net/socket.c:2460
#12 <signal handler called>
#13 0xb77b4424 in ?? ()
#14 0x00000002 in ?? ()
#15 0x00000001 in ?? ()
#16 0x0100007f in ?? ()
#17 0x00000000 in ?? ()
```

4

界定



Modules, or loadable modules, 在Linux系统中有特定含义，特制运行在内核空间中的可动态加载模块
很多内核模块的功能是设备驱动程序
有些设备驱动程序编译进内核，也有些运行在用户态

5

刘姥姥

main.c ✖ Makefile ✖

```
/*
 Example of a minimal character device driver
*/
#include <linux/module.h>

static int __init llaolao_init(void)
{
    int n = 0x1937;
    printk(KERN_INFO "Hi, I am llaolao at address 0x%p stack 0x%p.\n",
        llaolao_init, &n);

    return 0;
}

static void __exit llaolao_exit(void)
{
    printk("Exiting from 0x%p... Bye, GEDU friends\n", llaolao_exit);
}

module_init(llaolao_init);
module_exit(llaolao_exit);

MODULE_AUTHOR("GEDU lab");
MODULE_DESCRIPTION("LKM example - llaolao");
MODULE_LICENSE("GPL");
```

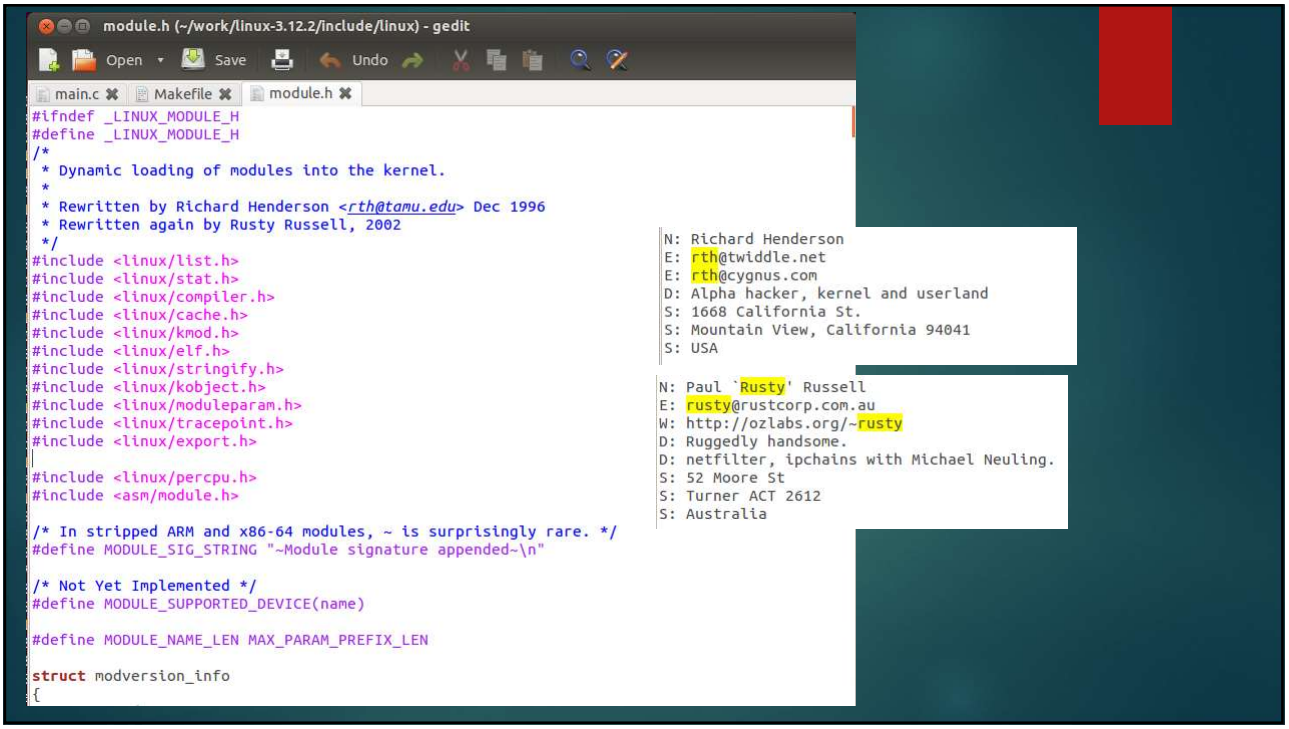
- Hello world
- 一进LINUX大观园
- A minimal LKM
- A bit more than that ☺

6

<linux/module.h>

- All Linux kernel modules must include <linux/module.h>
- Module.h defines a number of macros that are used by the kernel build system to add necessary metadata to compiled module files
- Defines the central 'struct module'

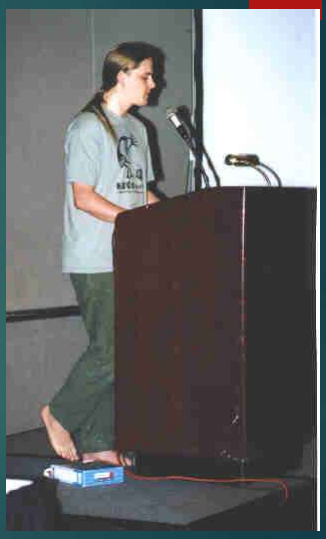
7



8

Richard Henderson

Shared Libraries from Ground Zero
In this talk I will describe ELF shared library internals and function across several architectures -- what makes for position independent code, the function of the PLT, and how the shared library loader bootstraps given that it is a shared library itself. Throughout, I will highlight interesting or troublesome points encountered while implementing ELF on the DEC Alpha.
Richard Henderson is a former student of Texas A&M University who, since graduation, has filled his spare time working on the Linux/Alpha compiler tool chain, sailing, and sampling the wares down at the pub.



9

10

Rusty Russell

- Rusty Russell is a Linux kernel hacker living in Australia, working for IBM's Linux Technology Center. He's also a frequent and talented speaker at Linux gatherings. When talking about Rusty in an earlier interview, Andrew Morton summarized, "he's just a really sharp and witty guy - anyone who has attended one of his sessions at a conference will attest to that!"
- Well known for his packet filtering efforts, having written both ipchains and netfilter/iptables, he has continued to make an impressive number of contributions to Linux kernel development. A large sampling of his current projects have been merged into the upcoming 2.6 kernel, including futexes, per-cpu counters, hot pluggable CPU support, and **a complete rewrite of the in-kernel module loading code.**
- For a humorous sample of Rusty's wit, one only needs to look to his email signature which reads, "Anyone who quotes me in their sig is an idiot. -- Rusty Russell." Read on for the full interview.

http://unixresources.net/linux/clf/linuxK/archive/00/00/46/02/460220.html

11

Rusty Russell

- http://www.elinux.org/Rusty_Russell_Quotes
- https://github.com/rustyrussell/
- https://rusty.ozlabs.org/

"I live in Canberra, Australia, married to a beautiful non-hacker, and expecting our first child in January. I have an engineering degree, was born in 1973, and I work for IBM's Linux Technology Center. "
-- **Interview: Rusty Russell Posted**: 2003-11-28 15:58

** Rusty and his daughter, from his blog in 2009*

12

Rusty Russell

Infrastructure Tech Engineer

Left IBM to join Blockstream in 2015

Rusty is a well-known Linux kernel maintainer, renowned for his foundational work on modules and iptables. After 14 years as a senior developer at IBM, he took a six-month sabbatical to work on cryptocurrencies. It was his top-notch work and writings during this period that brought him to our attention. Rusty has worked on a plethora of low-level code over the years, spoken at dozens of conferences around the world and also founded the first Australian Linux conference. He continues to maintain some Linux kernel parts as a hobby, while devoting most of his time to exploring the emerging frontier of Bitcoin development.

https://blockstream.com/team/rusty-russell/

13

LKM代码重构

JA: You rewrote the kernel module code in the Linux 2.5 development kernel. Can you explain a little about what you've done, and why you did it?

Rusty Russell: During 2.3, I was looking at trying to resolve a fairly tricky module unloading race with the ip_conntrack module. I had an idea, but it meant modifying the "struct module". Unfortunately, half the module loader was in userspace, so you can't just modify it, you have to be backward compatible, and test at runtime (and do what if it wasn't there?). This irritated me, and as I looked harder, I saw many other things caused by the module code's growth by accretion:

```
/*
* Originally by Anonymous (as far as I know...)
* Linux version by Bas Laarhoven
* 0.99.14 version by Jon Tombs ,
* Heavily modified by Bjorn Ekwall May 1994 (C)
* Rewritten by Richard Henderson Dec 1996
* Add MOD_INITIALIZING Keith Owens Nov 1999
```

Kernighan and Pike said "Don't patch bad code, rewrite it". So I wrote a prototype in-kernel module loader, and was a little surprised when it turned out to be less kernel code than the old code. The next step was to port it to more complex architectures than x86, to be fairly sure there was enough infrastructure. Along the way, I introduced new module parameter macros which also added boot parameters, sped up module reference counting so it's lockless and cpu-local, free parts of modules only used for initialization, added an option to disable module unloading, added forced module removal and blocking module removal, reimplemented modversions properly (with Kai Germaschewski), and tried really hard not to break any existing drivers which were not already broken. The downside is that the code is probably no longer smaller than it was with all these features turned on.

But the upside is that many of these changes occurred after the initial patch: but the userspace tools still work fine. You can now write a portable insmod in 20 lines.

** **Interview: Rusty Russell Posted**: 2003-11-28 15:58*

14

过程

JA: It sounds like a lot of improvements. How long have you been working on the module rewrite?

Rusty Russell: First reference I can find in my diary is May 2001, and I worked on it in spurts: it was something I did on the way to something else, but kinda became a full time task for about two months after the merge in November 2002, when I had to write a complete set of userspace utilities.

JA: When upgrading to the 2.6 kernel, one also has to upgrade to your new 'module-init-tools' to support these module changes. Are these utilities backward compatible, allowing one to switch between 2.4 and 2.6?

Rusty Russell: I wanted to keep the source trees completely separate, as they have different maintainers and on some architectures there are no "old utilities". I chose to have the new utilities simply exec the ".old" versions if they detected older kernels, and most people seem happy with this.

In some places, I chose not to be compatible, because I will have to maintain the result. Two things stand out to users: firstly my insmod is just a very, very dumb "insert this file in the kernel". I believe that insmod and rmmod are low-level utils for hackers and other control freaks, and modprobe should be used by system administrators and scripts.

The second change is the configuration file: I simply refused to support the old one. I believe that configuration formats should be minimal: if you want something more, then create a template that you can run through CPP, PHP or m4, and share it with the world. Rather than 32 directives, my config language has 5. There's a simple translator, of course, written in shell.

15

main.c | Makefile | module.h | CREDITS

```
 * 2.   So the community can ignore bug reports including proprietary modules
 * 3.   So vendors can do likewise based on their own policies
 */
#define MODULE_LICENSE(_license) MODULE_INFO(license, _license)

/*
 * Author(s), use "Name <email>" or just "Name", for multiple
 * authors use multiple MODULE_AUTHOR() statements/lines.
 */
#define MODULE_AUTHOR(_author) MODULE_INFO(author, _author)

/* What your module does. */
#define MODULE_DESCRIPTION(_description) MODULE_INFO(description, _description)

#define MODULE_DEVICE_TABLE(type,name)      \
  MODULE_GENERIC_TABLE(type##_device,name)

/* Version of form [<epoch>:]<version>[-<extra-version>].
   Or for CVS/RCS ID version, everything but the number is stripped.
  <epoch>: A (small) unsigned integer which allows you to start versions
           anew. If not mentioned, it's zero.  eg. "2:1.0" is after
        "1:2.0".
  <version>: The <version> may contain only alphanumerics and the
           character `.'.  Ordered by numeric sort for numeric parts,
        ascii sort for ascii parts (as per RPM or DEB algorithm).
  <extraversion>: Like <version>, but inserted for local
           customizations, eg "rh3" or "rusty1".

  Using this automatically adds a checksum of the .c files and the
  local headers in "srcversion".
*/

#if defined(MODULE) || !defined(CONFIG_SYSFS)
```

16

```
struct module
{
    enum module_state state;

    /* Member of list of modules */
    struct list_head list;

    /* Unique handle for this module */
    char name[MODULE_NAME_LEN];

    /* Sysfs stuff. */
    struct module_kobject mkobj;
    struct module_attribute *modinfo_attrs;
    const char *version;
    const char *srcversion;
    struct kobject *holders_dir;

    /* Exported symbols */
    const struct kernel_symbol *syms;
    const unsigned long *crcs;
    unsigned int num_syms;

    /* Kernel parameters. */
    struct kernel_param *kp;
    unsigned int num_kp;

    /* GPL-only exported symbols. */
    unsigned int num_gpl_syms;
    const struct kernel_symbol *gpl_syms;
    const unsigned long *gpl_crcs;
```

- 内核模块的“户籍档案”
- 颇为庞大的结构体
- 以链表形式相互关联，模块列表
- 只是LKM，不包括内核本身

17

https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/commit/?h=linux-3.18.y&id=c9e8928a

index : kernel/git/stable/linux-stable.git

Linux kernel stable tree

Greg Kroah-Hartman

about summary refs log tree commit diff stats

author Rusty Russell <rusty@rustcorp.com.au> 2016-02-03 06:25:26 (GMT)
committer Sasha Levin <sasha.levin@oracle.com> 2016-04-14 00:44:39 (GMT)
commit c9e8928aeeb09bfb7e647ac997c29ac3e6e5eeaa (patch)
tree 86498beb597ed12c1c53699b0906a846679104f8
parent 03f41fac326b77e5f5baf059f7f007677a1b70c8 (diff)

modules: fix longstanding /proc/kallsyms vs module insertion race.

[Upstream commit 8244062ef1e54502ef55f54cced659913f244c3e]

For CONFIG_KALLSYMS, we keep two symbol tables and two string tables. There's one full copy, marked SHF_ALLOC and laid out at the end of the module's init section. There's also a cut-down version that only contains core symbols and strings, and lives in the module's core section.

After module init (and before we free the module memory), we switch the mod->symtab, mod->num_symtab and mod->strtab to point to the core versions. We do this under the module_mutex.

However, kallsyms doesn't take the module_mutex: it uses preempt_disable() and rcu tricks to walk through the modules, because it's used in the oops path. It's also used in /proc/kallsyms. There's nothing atomic about the change of these variables, so we can get the old (larger!) num_symtab and the new symtab pointer; in fact this is what I saw when trying to reproduce.

By grouping these variables together, we can use a carefully-dereferenced pointer to ensure we always get one or the other (the free of the module init section is already done in an RCU callback, so that's safe). We allocate the init one at the end of the module init section, and keep the core one inside the struct module itself (it could also have been allocated at the end of the module core, but that's probably overkill).

还在持续改进

18

```
diff --git a/include/linux/module.h b/include/linux/module.h
index 71f282a..18edb96 100644
--- a/include/linux/module.h
+++ b/include/linux/module.h
@@ -224,6 +224,12 @@ struct module_ref {
        unsigned long decs;
 } __attribute((aligned(2 * sizeof(unsigned long))));

+struct mod_kallsyms {
+       Elf_Sym *symtab;
+       unsigned int num_symtab;
+       char *strtab;
+};
+
 struct module {
        enum module_state state;

@@ -311,14 +317,9 @@ struct module {
 #endif

 #ifdef CONFIG_KALLSYMS
-       /*
-        * We keep the symbol and string tables for kallsyms.
-        * The core_* fields below are temporary, loader-only (they
-        * could really be discarded after module init).
-        */
-       Elf_Sym *symtab, *core_symtab;
-       unsigned int num_symtab, core_num_syms;
-       char *strtab, *core_strtab;
+       /* Protected by RCU and/or module_mutex: use rcu_dereference() */
+       struct mod_kallsyms *kallsyms;
+       struct mod_kallsyms core_kallsyms;

        /* Section attributes */
        struct module_sect_attrs *sect_attrs;
```

"By grouping these variables together, we can use a carefully-dereferenced pointer to ensure we always get one or the other (the free of the module init section is already done in an RCU callback, so that's safe). We allocate the init one at the end of the module init section, and keep the core one inside the struct module itself (it could also have been allocated at the end of the module core, but that's probably overkill)."
-- Rusty Russell

19

<linux/init.h>

```
/* Each module must use one module_init(). */
#define module_init(initfn)                    \
        static inline initcall_t __inittest(void)  \
        { return initfn; }                     \
        int init_module(void) __attribute__((alias(#initfn)));

/* This is only required if you want to be unloadable. */
#define module_exit(exitfn)                    \
        static inline exitcall_t __exittest(void)  \
        { return exitfn; }                     \
        void cleanup_module(void) __attribute__((alias(#exitfn)));
```

```
module_init(llaolao_init);
module_exit(llaolao_exit);
```

20

Makefile

main.c ✖ Makefile ✖

```
WFLAGS := -Wall -Wstrict-prototypes -Wno-trigraphs

LDFLAGS = -Map /var/tmp/lllmap.txt

EXTRA_CFLAGS := $(WFLAGS)

# EXTRA_CFLAGS += -g -Wa,-adhln=$(<:.c=.lst)

EXTRA_CFLAGS += -D_DEBUG -g3   -fno-stack-protector

MODULE = llaolao

obj-m := $(MODULE).o

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

$(MODULE)-objs := main.o

clean:
    rm -rf *.o *~ .*.cmd *.ko *.mod.c *.order *.symvers .tmp_versions built-in.o
```

必须TAB不是空格

- 姑且用之，以后慢慢理解

$ make

```
ge@gewubox:~/work/llaolao$ make
make -C /lib/modules/3.12.2/build SUBDIRS=/home/ge/work/llaolao modules
make[1]: Entering directory `/home/ge/work/linux-3.12.2'
  CC [M]  /home/ge/work/llaolao/main.o
  LD [M]  /home/ge/work/llaolao/llaolao.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/ge/work/llaolao/llaolao.mod.o
  LD [M]  /home/ge/work/llaolao/llaolao.ko
make[1]: Leaving directory `/home/ge/work/linux-3.12.2'
```

- $ make –C /lib/modules/x.y.z/build SUBDIRS=<> modules

格物

```
ge@gewubox:~/work/llaolao$ ll
total 2404
drwxrwxr-x 3 ge ge   4096 Aug 15 21:32 ./
drwxrwxr-x 5 ge ge   4096 Aug 15 20:57 ../
-rw-rw-r-- 1 ge ge 946162 Aug 15 21:32 llaolao.ko
-rw-rw-r-- 1 ge ge    243 Aug 15 21:32 .llaolao.ko.cmd
-rw-rw-r-- 1 ge ge    702 Aug 15 21:32 llaolao.mod.c
-rw-rw-r-- 1 ge ge 477028 Aug 15 21:32 llaolao.mod.o
-rw-rw-r-- 1 ge ge  27567 Aug 15 21:32 .llaolao.mod.o.cmd
-rw-rw-r-- 1 ge ge 470118 Aug 15 21:32 llaolao.o
-rw-rw-r-- 1 ge ge    137 Aug 15 21:32 .llaolao.o.cmd
-rw-rw-r-- 1 ge ge    515 Aug 15 21:32 main.c
-rw-rw-r-- 1 ge ge 470116 Aug 15 21:32 main.o
-rw-rw-r-- 1 ge ge  27431 Aug 15 21:32 .main.o.cmd
-rw-rw-r-- 1 ge ge    467 Aug 15 21:19 Makefile
-rw-rw-r-- 1 ge ge     40 Aug 15 21:32 modules.order
-rw-rw-r-- 1 ge ge      0 Aug 15 21:32 Module.symvers
drwxrwxr-x 2 ge ge   4096 Aug 15 21:32 .tmp_versions/
```

- .o – 编译.c产生的object文件

23

.mod.c

```
ge@gewubox:~/work/llaolao$ cat llaolao.mod.c
#include <linux/module.h>
#include <linux/vermagic.h>
#include <linux/compiler.h>

MODULE_INFO(vermagic, VERMAGIC_STRING);

struct module __this_module
__attribute__((section(".gnu.linkonce.this_module"))) = {
        .name = KBUILD_MODNAME,
        .init = init_module,
#ifdef CONFIG_MODULE_UNLOAD
        .exit = cleanup_module,
#endif
        .arch = MODULE_ARCH_INIT,
};

static const struct modversion_info ____versions[]
__used
__attribute__((section("__versions"))) = {
        { 0xcddb328, __VMLINUX_SYMBOL_STR(module_layout) },
        { 0x50eedeb8, __VMLINUX_SYMBOL_STR(printk) },
};

static const char __module_depends[]
__used
__attribute__((section(".modinfo"))) =
"depends=";


MODULE_INFO(srcversion, "B4E598938FB071D2ED95D96");
```

- GCC工具链根据module.h中所定义宏内的编译指令（compiler directives）动态生成
- All symbols referenced by the module—together with the checksum that they need—are stored in the modversions_info array in the __modversions section. 加载模块时用以校验依赖性

24

check_version

```
kernel/module.c
static int check_version(Elf_Shdr *sechdrs,
                         unsigned int versindex,
                         const char *symname,
                         struct module *mod,
                         const unsigned long *crc)
{
        unsigned int i, num_versions;
        struct modversion_info *versions;

        /* Exporting module didn't supply crcs?  OK, we're already tainted. */
        if (!crc)
                return 1;
...
```

25

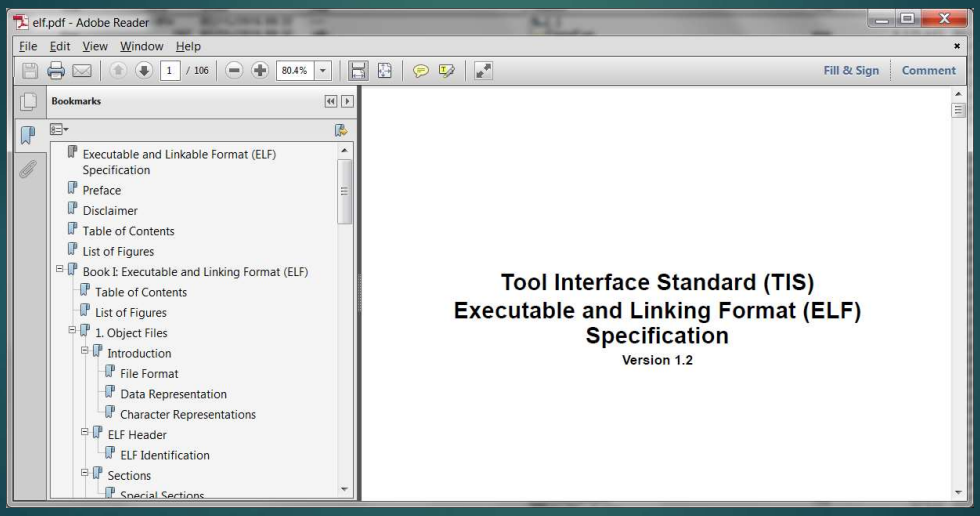
.ko

```
ge@gewubox:~/work/llaolao$ modinfo llaolao.ko
filename:       llaolao.ko
license:        GPL
description:    LKM example - llaolao
author:         GEDU lab
srcversion:     B4E598938FB071D2ED95D96
depends:
vermagic:       3.12.2 SMP mod_unload modversions 686
```

- Loadable kernel image file
- Linux世界的基础技术和标准件

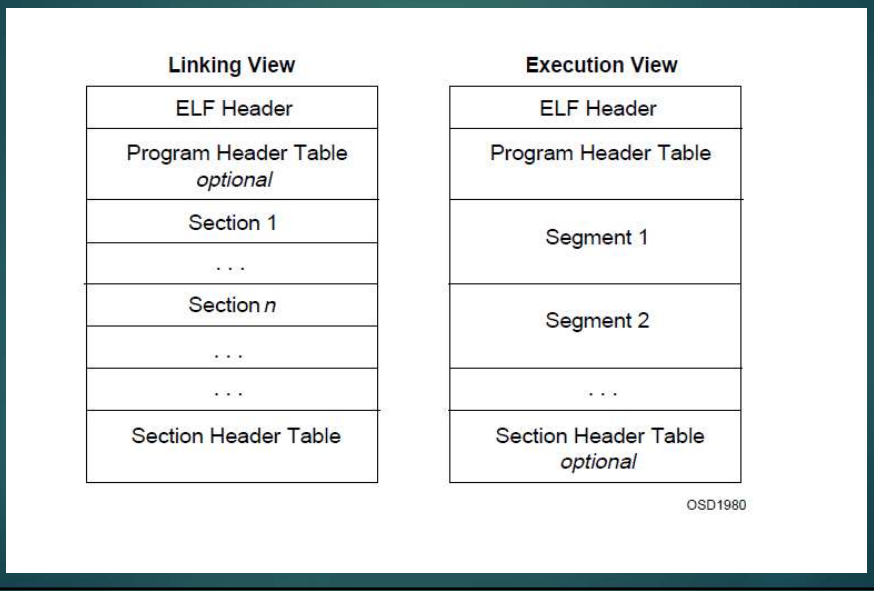
26

ELF



27

Layout



28

ELF Header

```
ge@gewubox:~/work/llaolao$ readelf -h llaolao.ko
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          930428 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           40 (bytes)
  Number of section headers:         34
  Section header string table index: 31
```

29

Section headers


```
ge@gewubox:~/work/llaolao$ readelf -S llaolao.ko
There are 34 section headers, starting at offset 0xe327c:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .note.gnu.build-i NOTE            00000000 000034 000024 00   A  0   0  4
  [ 2] .text             PROGBITS        00000000 000058 000000 00  AX  0   0  4
  [ 3] .exit.text        PROGBITS        00000000 000058 00001c 00  AX  0   0  1
  [ 4] .rel.exit.text    REL             00000000 0e37cc 000018 08      32   3  4
  [ 5] .init.text        PROGBITS        00000000 000074 00002c 00  AX  0   0  1
  [ 6] .rel.init.text    REL             00000000 0e37e4 000018 08      32   5  4
  [ 7] .rodata.str1.4    PROGBITS        00000000 0000a0 000058 01 AMS  0   0  4
  [ 8] .modinfo          PROGBITS        00000000 0000f8 00009a 00   A  0   0  1
  [ 9] __versions        PROGBITS        00000000 0001a0 000080 00   A  0   0 32
  [10] .data             PROGBITS        00000000 000220 000000 00  WA  0   0  4
  [11] .gnu.linkonce.thi PROGBITS        00000000 000220 000184 00  WA  0   0 32
  [12] .rel.gnu.linkonce REL             00000000 0e37fc 000010 08      32  11  4
  [13] .bss              NOBITS          00000000 0003a4 000000 00  WA  0   0  4
  [14] .comment          PROGBITS        00000000 0003a4 000056 01  MS  0   0  1
  [15] .note.GNU-stack   PROGBITS        00000000 0003fa 000000 00      0   0  1
  [16] .debug_aranges    PROGBITS        00000000 0003fa 000028 00      0   0  1
  [17] .rel.debug_arange REL             00000000 0e380c 000018 08      32  16  4
  [18] .debug_info       PROGBITS        00000000 000422 007d02 00      0   0  1
  [19] .rel.debug_info   REL             00000000 0e3824 0033c0 08      32  18  4
  [20] .debug_abbrev     PROGBITS        00000000 008124 00051e 00      0   0  1
  [21] .debug_line       PROGBITS        00000000 008642 00206d 00      0   0  1
  [22] .rel.debug_line   REL             00000000 0e6be4 000010 08      32  21  4
  [23] .debug_frame      PROGBITS        00000000 00a6b0 000054 00      0   0  4
  [24] .rel.debug_frame  REL             00000000 0e6bf4 000020 08      32  23  4
  [25] .debug_str        PROGBITS        00000000 00a704 00485a 01  MS  0   0  1
  [26] .debug_loc        PROGBITS        00000000 00ef5e 000070 00      0   0  1
  [27] .rel.debug_loc    REL             00000000 0e6c14 000080 08      32  26  4
  [28] .debug_macinfo    PROGBITS        00000000 00efce 0d4156 00      0   0  1
  [29] .debug_ranges     PROGBITS        00000000 0e3124 000018 00      0   0  1
  [30] .rel.debug_ranges REL             00000000 0e6c94 000020 08      32  29  4
  [31] .shstrtab         STRTAB          00000000 0e313c 00013e 00      0   0  1
  [32] .symtab           SYMTAB          00000000 0e6cb4 000250 10      33  33  4
  [33] .strtab           STRTAB          00000000 0e6f04 0000ee 00      0   0  1
Key to Flags:
```

30

加载和运行

```
ge@gewubox:~/work/llaolao$ sudo insmod llaolao.ko
```

```
insmod(8)                                                    insmod(8)

NAME
       insmod – simple program to insert a module into the Linux Kernel

SYNOPSIS
       insmod [filename]  [module options ...]

DESCRIPTION
       insmod  is  a  trivial program to insert a module into the kernel: if
       the filename is a hyphen, the module is taken  from  standard  input.
       Most users will want to use modprobe(8) instead, which is more clever
       and can handle module dependencies.

       Only the most general of error messages are reported: as the work  of
       trying  to  link  the module is now done inside the kernel, the dmesg
       usually gives more information about errors.
```

31

dmesg

[3343.073608] Hi, I am llaolao at address f083d000 stack d8973de4.

ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/sections/.init.text

0xf083d000

32

观察模块信息

```
ge@gewubox:~/work/llaolao$ ls -l /sys/module/llaolao
total 0
-r--r--r-- 1 root root 4096 Aug 17 22:31 coresize
drwxr-xr-x 2 root root    0 Aug 17 22:31 holders
-r--r--r-- 1 root root 4096 Aug 17 22:31 initsize
-r--r--r-- 1 root root 4096 Aug 17 22:31 initstate
drwxr-xr-x 2 root root    0 Aug 17 22:31 notes
-r--r--r-- 1 root root 4096 Aug 17 22:31 refcnt
drwxr-xr-x 2 root root    0 Aug 17 22:31 sections
-r--r--r-- 1 root root 4096 Aug 17 22:31 srcversion
-r--r--r-- 1 root root 4096 Aug 17 22:31 taint
--w------- 1 root root 4096 Aug 17 22:31 uevent
```

33

模块属性

```
ge@gewubox:~/work/llaolao$ cat /sys/module/llaolao/coresize
12395
ge@gewubox:~/work/llaolao$ cat /sys/module/llaolao/initsize
0
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/initsize
0
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/initstate
live
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/taint
OF
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/srcversion
B4E598938FB071D2ED95D96
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/refcnt
0
```

34

lsmod

- 列出所有加载模块
- "Module" denotes the name of the module.
- "Size" denotes the size of the module (not memory used).
- "Used by" denotes each module's use count and a list of referring modules. The "Used by" list is sometimes incomplete.
- If the module controls its own unloading via a can_unload routine then the use count displayed by lsmod is always -1, irrespective of the real use count.

```
Module                  Size  Used by
vboxsf                 38472  1
rfcomm                 57808  0
bnep                   18960  2
bluetooth             327609  10 rfcomm,bnep
parport_pc             31968  0
ppdev                  17363  0
snd_intel8x0           37241  2
snd_ac97_codec        109846  1 snd_intel8x0
ac97_bus               12642  1 snd_ac97_codec
snd_pcm                85369  2 snd_intel8x0,snd_ac97_codec
snd_seq_midi           13132  0
snd_rawmidi            25155  1 snd_seq_midi
crc32_pclmul           12967  0
snd_seq_midi_event     14475  1 snd_seq_midi
snd_seq                55403  2 snd_seq_midi,snd_seq_midi_event
vboxguest             238260  6 vboxsf
aesni_intel            18156  0
snd_timer              28639  2 snd_pcm,snd_seq
ablk_helper            13357  1 aesni_intel
cryptd                 15579  1 ablk_helper
snd_seq_device         14137  3 snd_seq_midi,snd_rawmidi,snd_seq
joydev                 17101  0
lrw                    13098  1 aesni_intel
snd                    60898  12 snd_intel8x0,snd_ac97_codec,snd_pcm,snd_seq_midi,snd_rawmidi,snd_seq,snd_time
psmouse                90880  0
aes_i586               16995  1 aesni_intel
xts                    12749  1 aesni_intel
gf128mul               14503  2 lrw,xts
soundcore              12600  1 snd
microcode              18928  0
i2c_piix4              17723  0
video                  18742  0
mac_hid                13037  0
snd_page_alloc         14230  2 snd_intel8x0,snd_pcm
serio_raw              13230  0
lp                     13299  0
parport                40803  3 parport_pc,ppdev,lp
hid_generic            12492  0
usbhid                 47035  0
hid                    87353  2 hid_generic,usbhid
ahci                   25579  2
libahci                26541  1 ahci
e1000                 128446  0
```

35

#modinfo

```
root@gewubox:/home/ge/work# modinfo e1000
filename:       /lib/modules/3.12.2/kernel/drivers/net/ethernet/intel/e1000/e1000.ko
version:        7.3.21-k8-NAPI
license:        GPL
description:    Intel(R) PRO/1000 Network Driver
author:         Intel Corporation, <linux.nics@intel.com>
srcversion:     FA229B930E96122838CC6B8
alias:          pci:v00008086d00002E6Esv*sd*bc*sc*i*
alias:          pci:v00008086d000010B5sv*sd*bc*sc*i*
...
alias:          pci:v00008086d00001000sv*sd*bc*sc*i*
depends:
intree:         Y
vermagic:       3.12.2 SMP mod_unload modversions 686
parm:           TxDescriptors:Number of transmit descriptors (array of int)
parm:           RxDescriptors:Number of receive descriptors (array of int)
parm:           Speed:Speed setting (array of int)
parm:           Duplex:Duplex setting (array of int)
parm:           AutoNeg:Advertised auto-negotiation setting (array of int)
parm:           FlowControl:Flow Control setting (array of int)
parm:           XsumRX:Disable or enable Receive Checksum offload (array of int)
parm:           TxIntDelay:Transmit Interrupt Delay (array of int)
parm:           TxAbsIntDelay:Transmit Absolute Interrupt Delay (array of int)
parm:           RxIntDelay:Receive Interrupt Delay (array of int)
parm:           RxAbsIntDelay:Receive Absolute Interrupt Delay (array of int)
parm:           InterruptThrottleRate:Interrupt Throttling Rate (array of int)
parm:           SmartPowerDownEnable:Enable PHY smart power down (array of int)
parm:           copybreak:Maximum size of packet that is copied to a new buffer on receive (uint)
parm:           debug:Debug level (0=none,...,16=all) (int)
```

36

卸载模块

```
$ sudo rmmod llaolao
```

```
[ 2911.263615] Exiting from 0xf0984000... Bye, GEDU friends
```

- module_exit函数会被调用

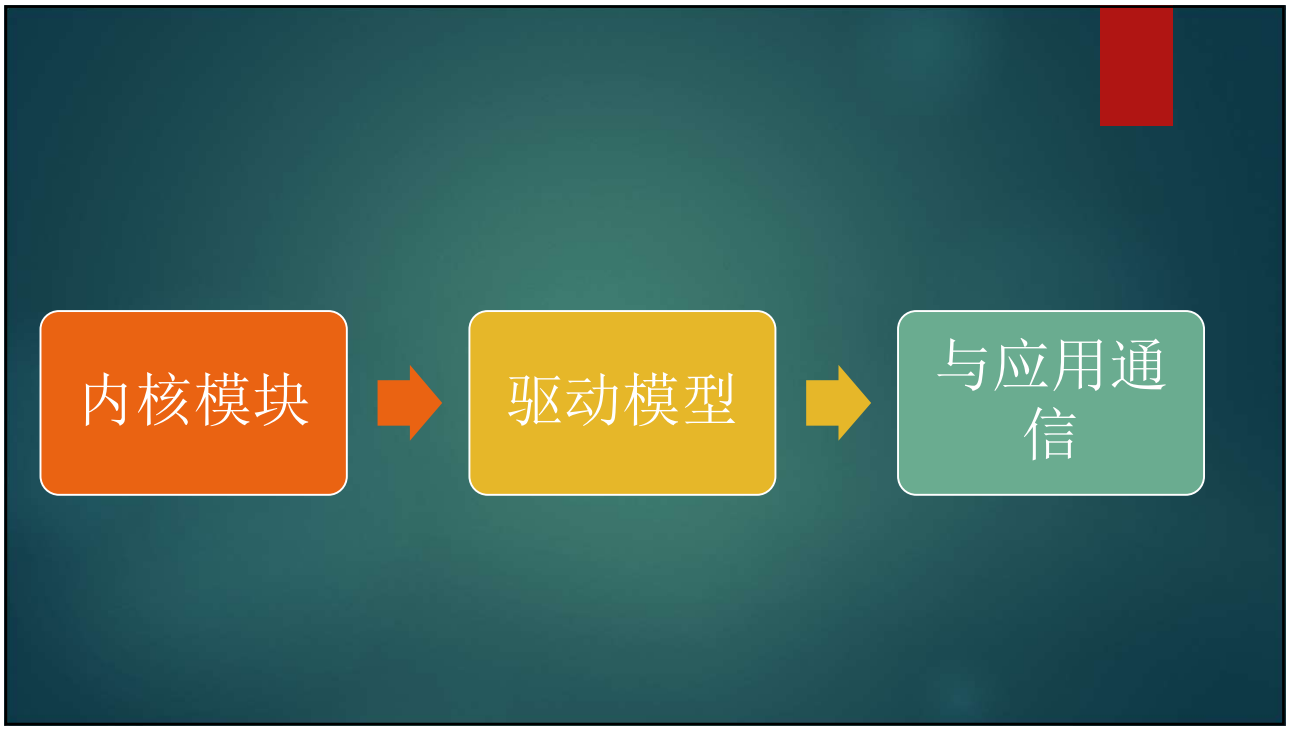
37

sections

```
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/sections/.
./                      .note.gnu.build-id
../                     .rodata.str1.4
.exit.text              .strtab
.gnu.linkonce.this_module  .symtab
.init.text
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/sections/.strtab
0xf0836250
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/sections/.symtab
0xf0836000
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/sections/.init.text
0xf0835000
ge@gewubox:~/work/llaolao$ sudo cat /sys/module/llaolao/sections/.gnu.linkonce
.this_module
0xf0981000
```

- 每个节在内存中的起始地址（线性地址）

38



39

The Linux Device Model

- The new Linux device model introduces C++-like abstractions that factor out commonalities from device drivers into bus and core layers
- user-space daemons and utilities
  - *udevd* and *udevinfo*
- User-specified rules located in the */etc/udev/rules.d/* directory

```
root@gewubox:/home/ge/work# ps -A | grep udevd
  306 ?        00:00:00 udevd
  553 ?        00:00:00 udevd
  554 ?        00:00:00 udevd
```

Documentation/driver-model/

d:\bench\linux-3.16.3\Documentation\driver-model*.*

↑Name	Ext	Size	Date
[..]		<DIR>	11/25/
binding	txt	3,634	09/18/
bus	txt	4,230	09/18/
class	txt	4,625	09/18/
design-patterns	txt	3,187	09/18/
device	txt	3,391	09/18/
devres	txt	9,090	09/18/
driver	txt	7,802	09/18/
overview	txt	4,881	09/18/
platform	txt	10,734	09/18/
porting	txt	13,402	09/18/

40

Lister - [D:\bench\linux-3.16.3\Documentation\driver-model\overview.txt]

File Edit Options Help

```
The Linux Kernel Device Model

Patrick Mochel  <mochel@digitalimplant.org>

Drafted 26 August 2002
Updated 31 January 2006


Overview
~~~~~~~~


The Linux Kernel Driver Model is a unification of all the disparate driver
models that were previously used in the kernel. It is intended to augment the
bus-specific drivers for bridges and devices by consolidating a set of data
and operations into globally accessible data structures.

Traditional driver models implemented some sort of tree-like structure
(sometimes just a list) for the devices they control. There wasn't any
uniformity across the different bus types.

The current driver model provides a common, uniform data model for describing
a bus and the devices that can appear under the bus. The unified bus
model includes a set of common attributes which all busses carry, and a set
of common callbacks, such as device discovery during bus probing, bus
shutdown, bus power management, etc.
```

41

patrick mochel

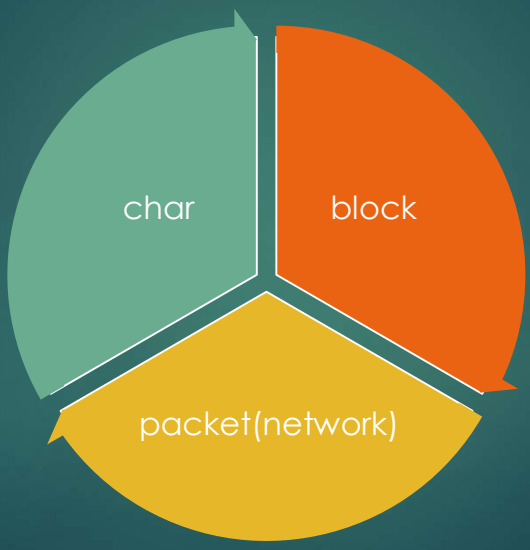
42

/drivers/base

```
/*
 * class.c - basic device class management
 *
 * Copyright (c) 2002-3 Patrick Mochel
 * Copyright (c) 2002-3 Open Source Development Labs
 * Copyright (c) 2003-2004 Greg Kroah-Hartman
 * Copyright (c) 2003-2004 IBM Corp.
 *
 * This file is released under the GPLv2
 *
 */
```

43

三大类



44

udevinfo

```
bash> udevinfo -a -p /sys/block/sr0
...
looking at the device chain at
'/sys/devices/pci0000:00/0000:00:1d.7/usb1/1-4':
 BUS=="usb"
 ID=="1-4"
 SYSFS{bConfigurationValue}=="1"
 ...
 SYSFS{idProduct}=="0701"
 SYSFS{idVendor}=="05e3"
 SYSFS{manufacturer}=="Genesyslogic"
 SYSFS{maxchild}=="0"
 SYSFS{product}=="USB Mass Storage Device"
```


Kobjects

- an encapsulation of common object properties such as usage reference counts.
- They are usually embedded within larger structures.
- Defined in include/linux/kobject.h:

/dev

- bash> ls -l /dev


dev.txt

```
ge@gewubox:~$ ls -l /dev
total 0
crw------T  1 root    root     10, 235 Aug 15 20:22 autofs
drwxr-xr-x  2 root    root          640 Aug 15 20:22 block
drwxr-xr-x  2 root    root          100 Aug 15 20:22 bsg
crw------T  1 root    root     10, 234 Aug 15 20:22 btrfs-control
drwxr-xr-x  3 root    root           60 Aug 15 20:23 bus
lrwxrwxrwx  1 root    root            3 Aug 15 20:22 cdrom -> sr0
lrwxrwxrwx  1 root    root            3 Aug 15 20:22 cdrom1 -> sr0
drwxr-xr-x  2 root    root         3460 Aug 15 20:23 char
crw-------  1 root    root      5,    1 Aug 15 20:22 console
lrwxrwxrwx  1 root    root           11 Aug 15 20:22 core -> /proc/kcore
drwxr-xr-x  2 root    root           60 Aug 15 20:22 cpu
crw-------  1 root    root     10,   60 Aug 15 20:22 cpu_dma_latency
drwxr-xr-x  6 root    root          120 Aug 15 20:22 disk
lrwxrwxrwx  1 root    root            3 Aug 15 20:22 dvd -> sr0
lrwxrwxrwx  1 root    root            3 Aug 15 20:22 dvd1 -> sr0
crw-------  1 root    root     10,   61 Aug 15 20:22 ecryptfs
lrwxrwxrwx  1 root    root           13 Aug 15 20:22 fd -> /proc/self/fd
crw-rw-rw-  1 root    root      1,    7 Aug 15 20:22 full
crw-rw-rwT  1 root    fuse     10, 229 Aug 15 20:22 fuse
crw-------  1 root    root    250,    0 Aug 15 20:22 hidraw0
crw-------  1 root    root     10, 228 Aug 15 20:22 hpet
drwxr-xr-x  4 root    root          320 Aug 15 20:22 input
crw-r--r--  1 root    root      1,   11 Aug 15 20:22 kmsg
srw-rw-rw-  1 root    root            0 Aug 15 20:22 log
brw-rw----  1 root    disk      7,    0 Aug 15 20:22 loop0
brw-rw----  1 root    disk      7,    1 Aug 15 20:22 loop1
brw-rw----  1 root    disk      7,    2 Aug 15 20:22 loop2
brw-rw----  1 root    disk      7,    3 Aug 15 20:22 loop3
brw-rw----  1 root    disk      7,    4 Aug 15 20:22 loop4
brw-rw----  1 root    disk      7,    5 Aug 15 20:22 loop5
brw-rw----  1 root    disk      7,    6 Aug 15 20:22 loop6
brw-rw----  1 root    disk      7,    7 Aug 15 20:22 loop7
crw-------  1 root    root     10, 237 Aug 15 20:22 loop-control
drwxr-xr-x  2 root    root           60 Aug 15 20:23 mapper
crw-------  1 root    root     10, 227 Aug 15 20:22 mcelog
crw-r-----  1 root    kmem      1,    1 Aug 15 20:22 mem
drwxr-xr-x  2 root    root           60 Aug 15 20:23 net
crw-------  1 root    root     10,   59 Aug 15 20:22 network_latency
```

47

Softirqs and Tasklets

- interrupt handlers have to exit as fast as possible
- interrupt handlers are designed in two parts: a hurried and harried top half that interacts with the hardware, and a relaxed bottom half that does most of the processing with all interrupts enabled
- Softirqs are the basic bottom half mechanism and have strong locking requirements
- Tasklets are built on top of softirqs and are easier to use

48

```
void __init
roller_init()
{
  /* ... */

  /* Open the softirq. Add an entry for ROLLER_SOFT_IRQ in
     the enum list in include/linux/interrupt.h */
  open_softirq(ROLLER_SOFT_IRQ, roller_analyze, NULL);
}

/* The interrupt handler */
static irqreturn_t
roller_interrupt(int irq, void *dev_id)
{
  /* Capture the wave stream */
  roller_capture();

  /* Mark softirq as pending */
  raise_softirq(ROLLER_SOFT_IRQ);

  return IRQ_HANDLED;
}
```

TOP HALF

```
/* The bottom half */
void
roller_analyze()
{
  /* Analyze the waveforms and switch to polled mode if required */
}
```

BOTTOM HALF

49

```
---Type <return> to continue, or q <return> to quit---
    at drivers/input/keyboard/atkbd.c:505
#13 0xc14fcb8c in serio_interrupt (serio=0xf6b29000, data=34 '"', dfl=<optimized out>) at drivers/input/serio/serio.c:985
#14 0xc14fdc37 in i8042_interrupt (irq=0, dev_id=<optimized out>)  at drivers/input/serio/i8042.c:535
#15 0xc10a6994 in handle_irq_event_percpu (desc=0xf68050c0, action=0xf6b1bf80) at kernel/irq/handle.c:142
#16 0xc10a6b7a in handle_irq_event (desc=0xf68050c0) at kernel/irq/handle.c:191
#17 0xc10a8f96 in handle_edge_irq (irq=<optimized out>, desc=0xf68050c0) at kernel/irq/chip.c:547
#18 0xc1012728 in execute_on_irq_stack (irq=1, desc=0xf6feace0, overflow=0) at arch/x86/kernel/irq_32.c:109
#19 handle_irq (irq=1, regs=<optimized out>) at arch/x86/kernel/irq_32.c:194
#20 0xc1683b72 in do_IRQ (regs=<optimized out>) at arch/x86/kernel/irq.c:193
#21 <signal handler called>
#22 0xc105ab0d in arch_local_irq_enable ()  at /home/ge/work/linux-3.12.2/arch/x86/include/asm/paravirt.h:819
#23 __do_softirq () at kernel/softirq.c:239
#24 0xc1012690 in call_on_stack (stack=0xc1949f0c, func=<optimized out>) at arch/x86/kernel/irq_32.c:71
#25 do_softirq () at arch/x86/kernel/irq_32.c:173
#26 0xc105ade6 in invoke_softirq () at kernel/softirq.c:340
#27 irq_exit () at kernel/softirq.c:374
#28 0xc1683c28 in exiting_irq () at /home/ge/work/linux-3.12.2/arch/x86/include/asm/apic.h:708
#29 smp_apic_timer_interrupt (regs=<optimized out>)  at arch/x86/kernel/apic/apic.c:930
#30 <signal handler called>
#31 native_safe_halt ()  at /home/ge/work/linux-3.12.2/arch/x86/include/asm/irqflags.h:50
#32 0xc1018041 in arch_safe_halt ()  at /home/ge/work/linux-3.12.2/arch/x86/include/asm/paravirt.h:111
#33 default_idle () at arch/x86/kernel/process.c:313
#34 0xc10187a6 in arch_cpu_idle () at arch/x86/kernel/process.c:302
#35 0xc10a5fe2 in cpu_idle_loop () at kernel/cpu/idle.c:98
#36 cpu_startup_entry (state=<optimized out>) at kernel/cpu/idle.c:133
#37 0xc166ba42 in rest_init () at init/main.c:392
#38 0xc19e6ac1 in start_kernel () at init/main.c:646
#39 0xc19e6396 in i386_start_kernel () at arch/x86/kernel/head32.c:49
#40 0x00000000 in ?? ()
```

50

常用结构体

Data Structure	Location	Description
tasklet_struct	include/linux/interrupt.h	Manages a tasklet, which is a method to implement bottom halves
kobject	include/linux/kobject.h	Encapsulates common properties of a kernel object
kset	include/linux/kobject.h	An object set to which a kobject belongs
kobj_type	include/linux/kobject.h	An object type that describes a kobject
class	include/linux/device.h	Abstracts the idea that a driver falls under a broader category
bus device device_driver	include/linux/device.h	Structures that form the pillars under the Linux device model

51

常用函数

Kernel Interface	Location	Description
request_irq()	kernel/irq/manage.c	Requests an IRQ and associates an interrupt handler with it
free_irq()	kernel/irq/manage.c	Frees an IRQ
disable_irq()	kernel/irq/manage.c	Disables the interrupt associated with a supplied IRQ
disable_irq_nosync()	kernel/irq/manage.c	Disables the interrupt associated with a supplied IRQ without waiting for any currently executing instances of the interrupt handler to return
enable_irq()	kernel/irq/manage.c	Re-enables the interrupt that has been disabled usingdisable_irq() ordisable_irq_nosync()
open_softirq()	kernel/softirq.c	Opens a softirq
raise_softirq()	kernel/softirq.c	Marks the softirq as pending execution
tasklet_init()	kernel/softirq.c	Dynamically initializes a tasklet
tasklet_schedule()	include/linux/interrupt.hkernel/softirq.c	Marks a tasklet as pending execution
tasklet_enable()	include/linux/interrupt.h	Enables a tasklet
tasklet_disable()	include/linux/interrupt.h	Disables a tasklet
tasklet_disable_nosync()	include/linux/interrupt.h	Disables a tasklet without waiting for active instances to finish execution
class_device_register() kobject_add() sysfs_create_dir() class_device_create() class_device_destroy() class_create() class_destroy() class_device_create_file() sysfs_create_file() class_device_add_attrs() kobject_uevent()	drivers/base/class.c lib/kobject.c lib/kobject_uevent.c fs/sysfs/dir.c fs/sysfs/file.c	Family of functions in the Linux device model that create/destroy a class, device class, and associated kobjects and sysfs files

52

Kernel Interface	Location	Description
alloc_chrdev_region()	fs/char_dev.c	Requests dynamic allocation of a device major number
unregister_chrdev_region()	fs/char_dev.c	Reverse ofalloc_chrdev_region()
cdev_init()	fs/char_dev.c	Connects char driver methods with the associated cdev
cdev_add()	fs/char_dev.c	Associates a device number with a cdev
cdev_del()	fs/char_dev.c	Removes a cdev
container_of()	include/linux/kernel.h	From a structure member, gets the address of its containing structure
copy_from_user()	arch/x86/lib/usercopy_32.c (For i386)	Copies data from user space to kernel space
copy_to_user()	arch/x86/lib/usercopy_32.c (For i386)	Copies data from kernel space to user space
likely() unlikely()	include/linux/compiler.h	Informs GCC about the possibility of success of the associated conditional evaluation
request_region()	include/linux/ioport.h kernel/resource.c	Stakes claim to an I/O region
release_region()	include/linux/ioport.h kernel/resource.c	Relinquishes claim to an I/O region
in[b | w | l | sn | sl]() out[b | w | l | sn | sl]()	include/asm-your-arch/io.h	Family of functions to exchange data with I/O regions
poll_wait()	include/linux/poll.h	Adds a wait queue to the kernel poll_table
fasync_helper()	fs/fcntl.c	Ensures that if a driver issues a kill_fasync(), a SIGIO is dispatched to the owning application
kill_fasync()	fs/fcntl.c	Dispatches a SIGIO to the owning application
parport_register_device()	drivers/parport/share.c	Registers a parallel port device with parport
parport_unregister_device()	drivers/parport/share.c	Unregisters a parallel port device
parport_register_driver()	drivers/parport/share.c	Registers a parallel port driver with parport
parport_unregister_driver()	drivers/parport/share.c	Unregisters a parallel port driver
parport_claim_or_block()	drivers/parport/share.c	Claims a parallel port
parport_write_data()	include/linux/parport.h	Writes data to a parallel port
parport_read_data()	include/linux/parport.h	Reads data from a parallel port
parport_release()	drivers/parport/share.c	Releases a parallel port
kobject_register()	lib/kobject.c	Registers a kobject and creates associated files in sysfs
kobject_unregister()	lib/kobject.c	Reverse ofkobject_register()
rtc_device_register()/ rtc_device_unregister()	drivers/rtc/class.c	Registers/unregisters a bottom-layer driver with the RTC subsystem
misc_register()	drivers/char/misc.c	Registers a misc driver
misc_deregister()	drivers/char/misc.c	Unregisters a misc driver

53

Functions of a character driver

- An initialization (or init()) routine that is responsible for initializing the device and seamlessly tying the driver to the rest of the kernel via registration functions.
- A set of entry points (or methods) such as open(), read(), ioctl(), llseek(), and write(), which directly correspond to I/O system calls invoked by user applications over the associated /dev node.
- Interrupt routines, bottom halves, timer handlers, helper kernel threads, and other support infrastructure. These are largely transparent to user applications.

54

Data structures of a character driver

- A per-device structure. This is the information repository around which the driver revolves.
- struct cdev, a kernel abstraction for character drivers. This structure is usually embedded inside the per-device structure referred previously.
- struct file_operations, which contains the addresses of all driver entry points.
- struct file, which contains information about the associated /dev node.

55

Device Example: System CMOS

- **CMOS** is short for **Complementary Metal-Oxide Semiconductor**.
- CMOS is an on-board, battery powered semiconductor chip inside computers that stores information

56

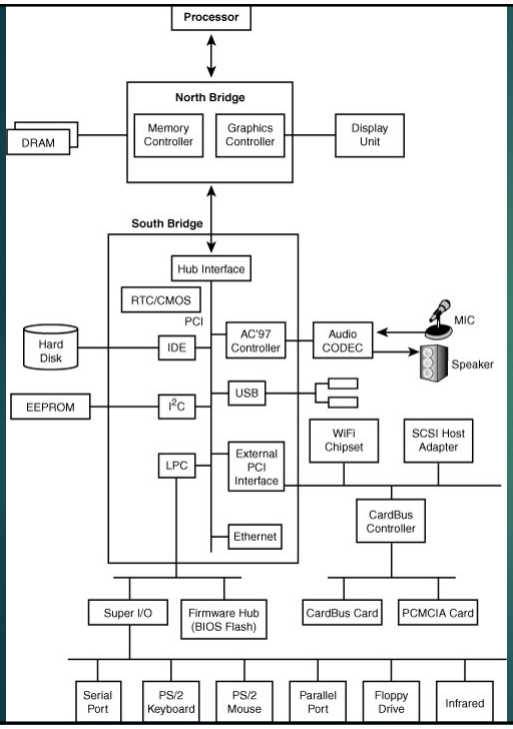
PC Ports

Lister - [F:\dig\industry\pc\ports.htm]

File Edit Options Help 22 %

```
          047 8254 Channel 3 control byte

<B>       060-067  8255 Programmable Peripheral Interface  (PC,XT, PCjr)
          060 8255 Port A keyboard input/output buffer (output PCjr)
          061 8255 Port B output
          062 8255 Port C input
          063 8255 Command/Mode control register

<B>       060-06F  8042 Keyboard Controller  (AT,PS/2)</B>
          060 8042 Keyboard input/output buffer register
          061 8042 system control port (for compatability with 8255)
          064 8042 Keyboard command/status register

          070 CMOS RAM/RTC, also NMI enable/disable (AT,PS/2, see <A hre
          071 CMOS RAM data  (AT,PS/2)

          080 Manufacturer systems checkpoint port (used during POST)
<B>       080-090  DMA Page Registers</B>
          081 High order 4 bits of DMA channel 2 address
          082 High order 4 bits of DMA channel 3 address
```

57

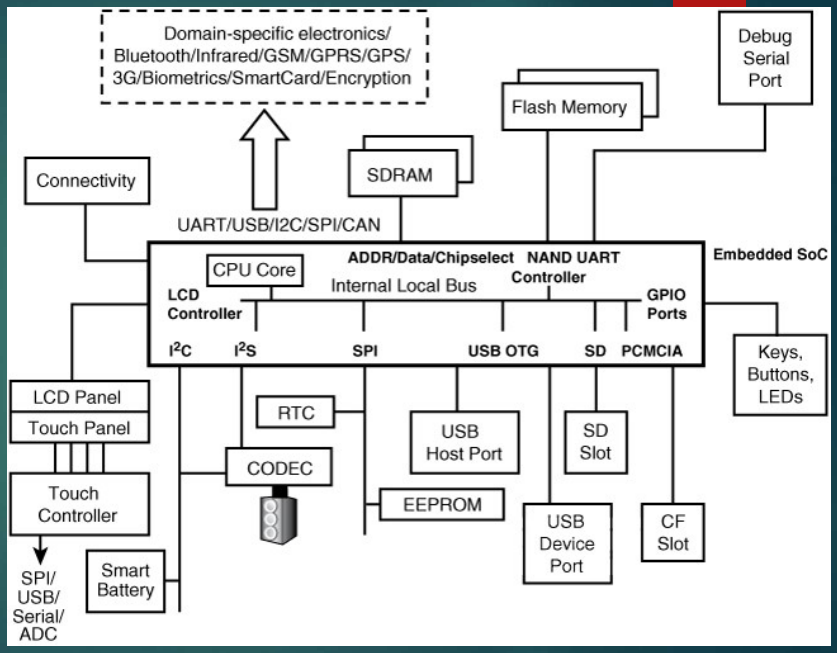


经典PC架构

- 处理器
- 北桥
- 南桥

58

典型SOC架构

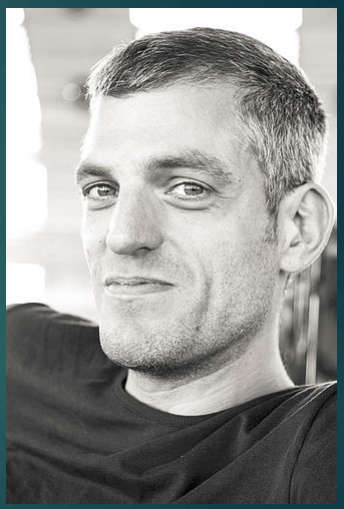


59

udevd

- 运行在用户空间的PnP服务
- 核心设计者：Kay Sievers，Red Hat
- 在用户空间实施PnP任务
  - 遍历rules.d下的规则文件
- /etc/rules.d
- 执行modprobe安装驱动

```
root@gedu-VirtualBox:/etc/udev# ls
hwdb.d  rules.d  udev.conf
```



60

内核模块 → 驱动模型 → 与应用通信

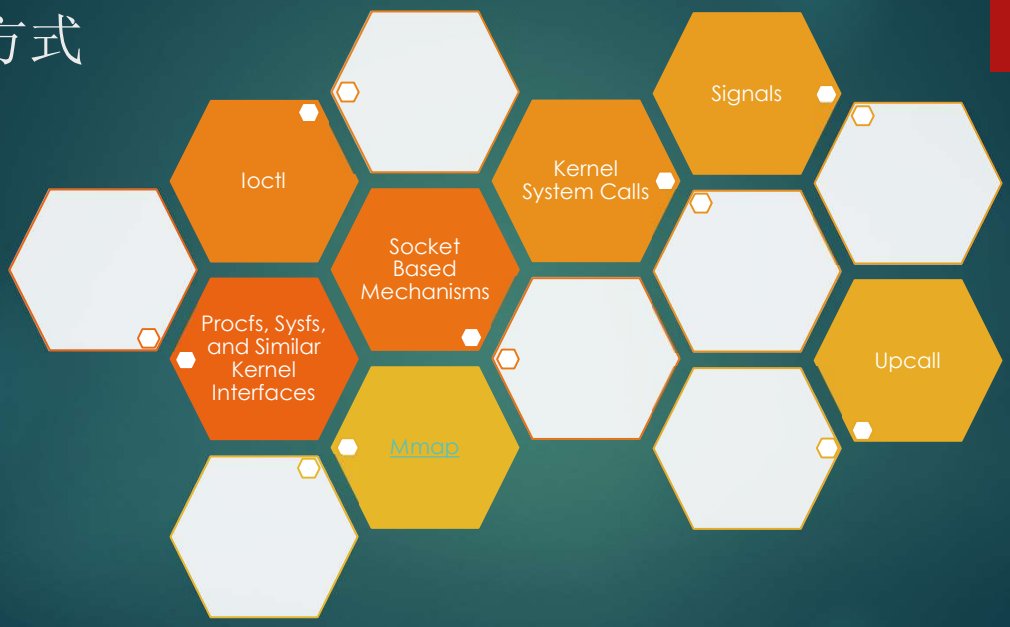
61

“此老生之常谭”

- Communication mechanisms between Linux user and kernel space
- 所有操作系统都有的问题

62

方式



http://wiki.tldp.org/kernel_user_space_howto

63

‘花灯’Device

- Upgrade llaolao to be a character driver managing series of Huadeng devices
- Demonstrates:
  - class_create()
  - cdev_init()
  - cdev_add()
  - device_create()
  - Communicate with user mode app via files under /dev

64

Step 1: 定义设备结构体

```
struct huadeng_dev {
  unsigned short current_pointer; /* Current pointer within the
                                     device data region */
  unsigned int size;              /* Size of the bank */
  int number;                     /* device number */
  struct cdev cdev;               /* The cdev structure */
  char name[10];                  /* Name of the device */
  char data[DEV_DATA_LENGTH];
  /* ... */                       /* Mutexes, spinlocks, wait
                                     queues, .. */
} *hd_devp[NUM_DEVICES];
```

65

Step 2: 创建设备类

```
#define DEVICE_NAME      "huadeng"
#define MAJOR_NUM        88

static struct class *huadeng_class;
```

huadeng_class = class_create(THIS_MODULE, DEVICE_NAME);

```
ge@gewubox:~/work/llaolao3$ ls -l /sys/class/huadeng
total 0
lrwxrwxrwx 1 root root 0 Sep 19 18:25 huadeng0 -> ../../devices/virtual/huadeng/
huadeng0
lrwxrwxrwx 1 root root 0 Sep 19 18:25 huadeng1 -> ../../devices/virtual/huadeng/
huadeng1
lrwxrwxrwx 1 root root 0 Sep 19 18:25 huadeng2 -> ../../devices/virtual/huadeng/
huadeng2
lrwxrwxrwx 1 root root 0 Sep 19 18:25 huadeng3 -> ../../devices/virtual/huadeng/
huadeng3
lrwxrwxrwx 1 root root 0 Sep 19 18:25 huadeng4 -> ../../devices/virtual/huadeng/
huadeng4
lrwxrwxrwx 1 root root 0 Sep 19 18:25 huadeng5 -> ../../devices/virtual/huadeng/
huadeng5
```

{ 0x116fab6d, __VMLINUX_SYMBOL_STR(__class_create) }

- to create the entry /sys/class/huadeng for use with the dynamic device registration mechanisms

66

```
struct class *__class_create(struct module *owner, const char *name,
                             struct lock_class_key *key)
{
        struct class *cls;
        int retval;

        cls = kzalloc(sizeof(*cls), GFP_KERNEL);
        if (!cls) {
                retval = -ENOMEM;
                goto error;
        }

        cls->name = name;
        cls->owner = owner;
        cls->class_release = class_create_release;

        retval = __class_register(cls, key);
        if (retval)
                goto error;

        return cls;

error:
        kfree(cls);
        return ERR_PTR(retval);
}
EXPORT_SYMBOL_GPL(__class_create);
```

```
int __class_register(struct class *cls, struct lock_class_key *key)
{
        struct subsys_private *cp;
        int error;

        pr_debug("device class '%s': registering\n", cls->name);

        cp = kzalloc(sizeof(*cp), GFP_KERNEL);
        if (!cp)
                return -ENOMEM;
        klist_init(&cp->klist_devices, klist_class_dev_get, klist_class_de
        INIT_LIST_HEAD(&cp->interfaces);
        kset_init(&cp->glue_dirs);
        __mutex_init(&cp->mutex, "subsys mutex", key);
        error = kobject_set_name(&cp->subsys.kobj, "%s", cls->name);
        if (error) {
                kfree(cp);
                return error;
        }

        /* set the default /sys/dev directory for devices of this class */
        if (!cls->dev_kobj)
                cls->dev_kobj = sysfs_dev_char_kobj;

#if defined(CONFIG_BLOCK)
        /* let the block class directory show up in the root of sysfs */
        if (!sysfs_deprecated || cls != &block_class)
                cp->subsys.kobj.kset = class_kset;
#else
        cp->subsys.kobj.kset = class_kset;
```


Step 3.1: 分配设备结构体实例

```
/* Allocate memory for the per-device structure */
hd_devp[i] = kmalloc(sizeof(struct huadeng_dev), GFP_KERNEL);
if (!hd_devp[i]) {
    printk("allocate huadeng dev struct failed\n");
    return -ENOMEM;
}
```

Step 3.2: 关联文件操作

```
struct file_operations huadeng_fops = {
  .owner = THIS_MODULE,
  .open = huadeng_open,
  .release = huadeng_release,
  .read = huadeng_read,
  .write = huadeng_write,
};
```

```
/* Connect the file operations with the cdev */
cdev_init(&hd_devp[i]->cdev, &huadeng_fops);
hd_devp[i]->cdev.owner = THIS_MODULE;
```

69

Step 3.3: 创建/dev节点

```
/* Send uevents to udev, so it'll create /dev nodes */
device_create(huadeng_class, NULL,
    MKDEV(MAJOR_NUM, i), NULL, "huadeng%d", i);
```

70

/drivers/base/core.c

```
/**
 * device_create - creates a device and registers it with sysfs
 * @class: pointer to the struct class that this device should be registered to
 * @parent: pointer to the parent struct device of this new device, if any
 * @devt: the dev_t for the char device to be added
 * @drvdata: the data to be added to the device for callbacks
 * @fmt: string for the device's name

struct device *device_create(struct class *class, struct device *parent,
                             dev_t devt, void *drvdata, const char *fmt, ...)
{
        va_list vargs;
        struct device *dev;

        va_start(vargs, fmt);
        dev = device_create_vargs(class, parent, devt, drvdata, fmt, vargs);
        va_end(vargs);
        return dev;
}
EXPORT_SYMBOL_GPL(device_create);
```

71

/dev

```
ge@gewubox:~/work/llaolao3$ ls -l /dev/huadeng*
crw------- 1 root root 88, 0 Sep 18 21:06 /dev/huadeng0
crw------- 1 root root 88, 1 Sep 18 21:06 /dev/huadeng1
crw------- 1 root root 88, 2 Sep 18 21:06 /dev/huadeng2
crw------- 1 root root 88, 3 Sep 18 21:06 /dev/huadeng3
crw------- 1 root root 88, 4 Sep 18 21:06 /dev/huadeng4
crw------- 1 root root 88, 5 Sep 18 21:06 /dev/huadeng5
```

72

ll

```
ge@gewubox:~/work/llaolao3$ ls -l /dev
total 0
crw------T  1 root    root     10, 235 Sep 18 19:53 autofs
drwxr-xr-x  2 root    root          640 Sep 18 19:53 block
drwxr-xr-x  2 root    root          100 Sep 18 19:53 bsg
crw------T  1 root    root     10, 234 Sep 18 19:53 btrfs-control
drwxr-xr-x  3 root    root           60 Sep 18 19:53 bus
lrwxrwxrwx  1 root    root            3 Sep 18 19:53 cdrom -> sr0
lrwxrwxrwx  1 root    root            3 Sep 18 19:53 cdrom1 -> sr0
drwxr-xr-x  2 root    root         3460 Sep 18 19:53 char
crw-------  1 root    root      5,    1 Sep 18 19:53 console
lrwxrwxrwx  1 root    root           11 Sep 18 19:53 core -> /proc/kcore
drwxr-xr-x  2 root    root           60 Sep 18 19:53 cpu
crw-------  1 root    root     10,   60 Sep 18 19:53 cpu_dma_latency
drwxr-xr-x  6 root    root          120 Sep 18 19:53 disk
lrwxrwxrwx  1 root    root            3 Sep 18 19:53 dvd -> sr0
lrwxrwxrwx  1 root    root            3 Sep 18 19:53 dvd1 -> sr0
crw-------  1 root    root     10,   61 Sep 18 19:53 ecryptfs
lrwxrwxrwx  1 root    root           13 Sep 18 19:53 fd -> /proc/self/fd
crw-rw-rw-  1 root    root      1,    7 Sep 18 19:53 full
crw-rw-rwT  1 root    fuse     10, 229 Sep 18 19:53 fuse
```

- First character:
  - c signifies a char driver, b stands for a block driver, d stands for directory, and l denotes a symbolic link
- 5th column is major no, 6th is minor no

73

Open

```
static int huadeng_open(struct inode *inode, struct file *file)
{
  struct huadeng_dev *hd_devp;
  pr_info("%s\n", __func__);

  /* Get the per-device structure that contains this cdev */
  hd_devp = container_of(inode->i_cdev, struct huadeng_dev, cdev);

  /* Easy access to hd_devp from rest of the entry points */
  file->private_data = hd_devp;

  /* Initialize some fields */
  hd_devp->size = DEV_DATA_LENGTH;
  hd_devp->current_pointer = 0;

  return 0;
}
```

74

```
static ssize_t huadeng_read(struct file *file,
  char *buffer, size_t count, loff_t * offset)
{
  struct huadeng_dev *hd_devp = file->private_data;
  pr_info("%s count %u, +%llu\n", __func__, count, *offset);

  if (*offset >= hd_devp->size) {
    return 0; /*EOF*/
  }
  /* Adjust count if its edges past the end of the data region */
  if (*offset + count > hd_devp->size) {
    count = hd_devp->size - *offset;
  }
  /* Copy the read bits to the user buffer */
  if (copy_to_user(buffer, (void *)(hd_devp->data + *offset), count) != 0) {
    return -EIO;
  }

  *offset += count;

  return count;
}
```

75

```
static ssize_t huadeng_write(struct file *file,
  const char *buffer, size_t length, loff_t * offset)
{
  struct huadeng_dev *hd_devp = file->private_data;
  pr_info("%s %u +%llx\n", __func__, length, *offset);
  if(*offset >= hd_devp->size) {
    return 0;
  }
  if (*offset + length > hd_devp->size) {
    length = hd_devp->size - *offset;
  }
  if(copy_from_user(hd_devp->data+*offset, buffer, length) !=0 ) {
    printk(KERN_ERR "copy_from_user failed\n");
    return -EFAULT;
  }

  return length;
}
```

76

应用程序baner

- 用户空间代码
- 通过文件与内核态驱动通信（交换数据）
- 王板儿，中国古典文学名著《红楼梦》里刘姥姥的外孙


```
int usage()
{
    printf("baner <devno> r/w/s/i <value>\n");
    return -1;
}
```

77

```
int
main(int argc, char * argv[])
{
    int n, fd;
    char name[20], buffer[20];
    if(argc < 3)
        return usage();

    sprintf(name, "/dev/huadeng%s", argv[1]);

    fd = open(name, O_RDWR);
    if(fd < 0)
    {
        printf("open file %s failed with %d\n", name, errno);
        return -2;
    }
    if(argv[2][0] == 'r')
    {
        while((n = read(fd, buffer, sizeof(buffer)))>0)
        {
            printf("read %d bytes:\n %s\n", n, buffer);
        };
    }
    else if(argv[2][0] == 'w')
    {
        if(argc < 4)
        {
            printf("missing what to write\n");
            goto tag_close;
        }

        if((n = write(fd, argv[3], strlen(argv[3])))>0)
        {
            printf("write %d bytes:\n %s\n", n, argv[3]);
        }
        else
            printf("write file %s failed %d\n", name, errno);
    }

tag_close:
    close(fd);
    return 0;
}
```

78

编译和运行

- # gcc –o baner baner.c
- # sudo ./baner 0 w goldfish
- #sudo ./baner 0 r

```
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 w goldfish
write 8 bytes:
 goldfish
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 r
read 20 bytes:
 goldfish
read 20 bytes:

read 20 bytes:

read 20 bytes:

read 20 bytes:
```

```
[ 7279.576809] huadeng_open
[ 7279.576825] huadeng_write 8 +0
[ 7279.579371] huadeng_release
[ 7283.620767] huadeng_open
[ 7283.620783] huadeng_read count 20, +0x0
[ 7283.622188] huadeng_read count 20, +0x20
[ 7283.622470] huadeng_read count 20, +0x40
[ 7283.622793] huadeng_read count 20, +0x60
[ 7283.623067] huadeng_read count 20, +0x80
[ 7283.623582] huadeng_read count 20, +0x100
[ 7283.623644] huadeng_release
```


od

- od - dump files in octal and other formats

```
SYNOPSIS
       od [OPTION]... [FILE]...
       od [-abcdfilosx]... [FILE] [[+]OFFSET[.][b]]
       od --traditional [OPTION]... [FILE] [[+]OFFSET[.][b] [+][LABEL][.][b]]

DESCRIPTION
       Write an unambiguous representation, octal bytes by default, of FILE to
       standard output.  With more than one FILE argument, concatenate them in
       the  listed  order to form the input.  With no FILE, or when FILE is -,
       read standard input.

       All arguments to long options are mandatory for short options.

       -A, --address-radix=RADIX
              decide how file offsets are printed

       -j, --skip-bytes=BYTES
              skip BYTES input bytes first

       -N, --read-bytes=BYTES
              limit dump to BYTES input bytes
```

经典格式（类WinDbg）

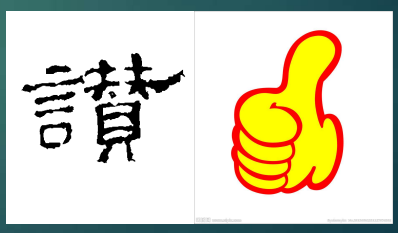
```
ge@gewubox:~/work/cmos$ sudo od --traditional -Ax -t x4 -N0x40 /dev/port +0x60
000060 ffff311c ffffff1c ffffffff ffffffff
000070 00ff02ff ffffffff ffffffff ffffffff
000080 00000000 00000000 00000000 00000000
000090 ff02ffff ffffffff ffffffff ffffffff
0000a0
```

- -Ax 第一列中的地址以十六进制显示
- -t x4 显示格式为4字节（DWORD）16进制
- -N0x40 显示的总字节数
- +0x60 起始的文件偏移

81

*

```
ge@gewubox:~/work/llaolao3$ sudo od --traditional -Ax -t x4 -N0x40 /dev/zero +0x
60
[sudo] password for ge:
000060 00000000 00000000 00000000 00000000
*
0000a0
```



82

I/O Control(ioctl)

- perform various types of hardware control via the device driver


```
int (*ioctl)  (struct inode *inode, struct file *filp,
               unsigned int cmd, unsigned long arg);
```


Ioctl Command

_IO	an ioctl with no parameters
_IOW	an ioctl with write parameters (copy_from_user)
_IOR	an ioctl with read parameters (copy_to_user)
_IOWR	an ioctl with both write and read parameters.

```
#define HUADENG_IOC_MAGIC 'o'

/*
 * S means "Set" through a ptr,
 * T means "Tell" directly with the argument value
 * G means "Get": reply by setting through a pointer
 * Q means "Query": response is on the return value
 * X means "eXchange": switch G and S atomically
 * H means "sHift": switch T and Q atomically
 */
#define HUADENG_IOCRESET      _IO(HUADENG_IOC_MAGIC, 0)
#define HUADENG_IOCSLUMINANCE _IOW(HUADENG_IOC_MAGIC,  1, int)
#define HUADENG_IOCTLUMINANCE _IO(HUADENG_IOC_MAGIC,   2)
#define HUADENG_IOCGLUMINANCE _IOR(HUADENG_IOC_MAGIC,  3, int)
#define HUADENG_IOCQLUMINANCE _IO(HUADENG_IOC_MAGIC,   4)
#define HUADENG_IOCXLUMINANCE _IOWR(HUADENG_IOC_MAGIC, 5, int)
#define HUADENG_IOCHLUMINANCE _IO(HUADENG_IOC_MAGIC,  6)
```

85

Using the ioctl Argument

- If it is an integer, it's easy: it can be used directly. If it is a pointer, however, some care must be taken
- *copy_from_user* and *copy_to_user* functions
- *access_ok*
- put_user(datum, ptr)
- _ _put_user(datum, ptr)
- get_user(local, ptr)
- _ _get_user(local, ptr)

86

access_ok

- int access_ok(int type, const void *addr, unsigned long size);
- VERIFY_READ or VERIFY_WRITE
- returns a boolean value: 1 for success (access is OK) and 0 for failure (access is not OK)

87

```
/*
 * These are the main single-value transfer routines.  They automatically
 * use the right size if we just have the right pointer type.
 * This version just falls back to copy_{from,to}_user, which should
 * provide a fast-path for small values.
 */
#define __put_user(x, ptr) \
({                                                      \
        __typeof__(*(ptr)) __x = (x);                   \
        int __pu_err = -EFAULT;                         \
        __chk_user_ptr(ptr);                            \
        switch (sizeof (*(ptr))) {                      \
        case 1:                                         \
        case 2:                                         \
        case 4:                                         \
        case 8:                                         \
                __pu_err = __put_user_fn(sizeof (*(ptr)),       \
                                         ptr, &__x);            \
                break;                                  \
        default:                                        \
                __put_user_bad();                       \
                break;                                  \
         }                                              \
        __pu_err;                                       \
})

#define put_user(x, ptr)                                \
({                                                      \
        might_fault();                                  \
        access_ok(VERIFY_WRITE, ptr, sizeof(*ptr)) ?    \
                __put_user(x, ptr) :                    \
                -EFAULT;                                \
})

#ifndef __put_user_fn
```

88

权限控制

- int capable(int capability);

```
if (! capable (CAP_SYS_ADMIN))
    return -EPERM;
```

CAP_DAC_OVERRIDE
The ability to override access restrictions (data access control, or DAC) on files and directories.
CAP_NET_ADMIN
The ability to perform network administration tasks, including those that affect network interfaces.
CAP_SYS_MODULE
The ability to load or remove kernel modules.
CAP_SYS_RAWIO
The ability to perform "raw" I/O operations. Examples include accessing device ports or communicating directly with USB devices.
CAP_SYS_ADMIN
A catch-all capability that provides access to many system administration operations.
CAP_SYS_TTY_CONFIG
The ability to perform tty configuration tasks.

89

```
static long
huadeng_ioctl(/*struct inode *inode,*/ struct file *file,
        unsigned int cmd, unsigned long arg)
{
    long retval = 0;
    int tmp = 0;
    struct huadeng_dev *hd_devp = file->private_data;
    pr_info("%s cmd 0x%x arg %lx, luminance %d\n",
        __func__, cmd, arg, hd_devp->luminance);

    switch(cmd) {

      case HUADENG_IOCRESET:
        hd_devp->luminance = HUADENG_LUMINANCE;
        break;

      case HUADENG_IOCSLUMINANCE: /* Set: arg points to the value */
        if (! capable (CAP_SYS_ADMIN))
            return -EPERM;
        retval = __get_user(hd_devp->luminance, (int __user *)arg);
        break;
```

90

```
case HUADENG_IOCTLUMINANCE: /* Tell: arg is the value */
      if (! capable (CAP_SYS_ADMIN))
         return -EPERM;
      hd_devp->luminance = arg;
      break;

    case HUADENG_IOCGLUMINANCE: /* Get: arg is pointer to result */
      retval = __put_user(hd_devp->luminance, (int __user *)arg);
      break;

    case HUADENG_IOCQLUMINANCE: /* Query: return it (it's positive) */
      return hd_devp->luminance;

    case HUADENG_IOCXLUMINANCE: /* eXchange: use arg as pointer */
      if (! capable (CAP_SYS_ADMIN))
         return -EPERM;
      tmp = hd_devp->luminance;
      retval = __get_user(hd_devp->luminance, (int __user *)arg);
      if (retval == 0)
         retval = __put_user(tmp, (int __user *)arg);
      break;
```

91

```
            case HUADENG_IOCHLUMINANCE: /* sHift: like Tell + Query */
              if (! capable (CAP_SYS_ADMIN))
                 return -EPERM;
              tmp = hd_devp->luminance;
              hd_devp->luminance = arg;
              return tmp;

            default:  /* redundant, as cmd was checked against MAXNR */
              return -ENOTTY;
          }
          return retval;
        }
```

92

```
void hd_ioctl(int fd, char * cmd, char *arg)
{
    int val = 0;
    switch(cmd[0])
    {
    case 's':
        val = atoi(arg);
        ioctl(fd, HUADENG_IOCSLUMINANCE, &val);
        break;
    case 't':
        val = atoi(arg);
        ioctl(fd, HUADENG_IOCTLUMINANCE, val);
        break;
    case 'g':
        ioctl(fd, HUADENG_IOCGLUMINANCE, &val);
        break;
    case 'q':
        val = ioctl(fd, HUADENG_IOCQLUMINANCE, 0);
        break;
    case 'x':
        val = atoi(arg);
        ioctl(fd, HUADENG_IOCXLUMINANCE, &val);
        break;
    case 'h':
        val = atoi(arg);
        val = ioctl(fd, HUADENG_IOCHLUMINANCE, val);
        break;
    case 'r':
        ioctl(fd, HUADENG_IOCRESET, 0);
        break;
    default:
        printf("unsupported cmd %s\n", cmd);
        break;
    }
    printf("ioctl cmd %s value %d\n", cmd, val);
}
```

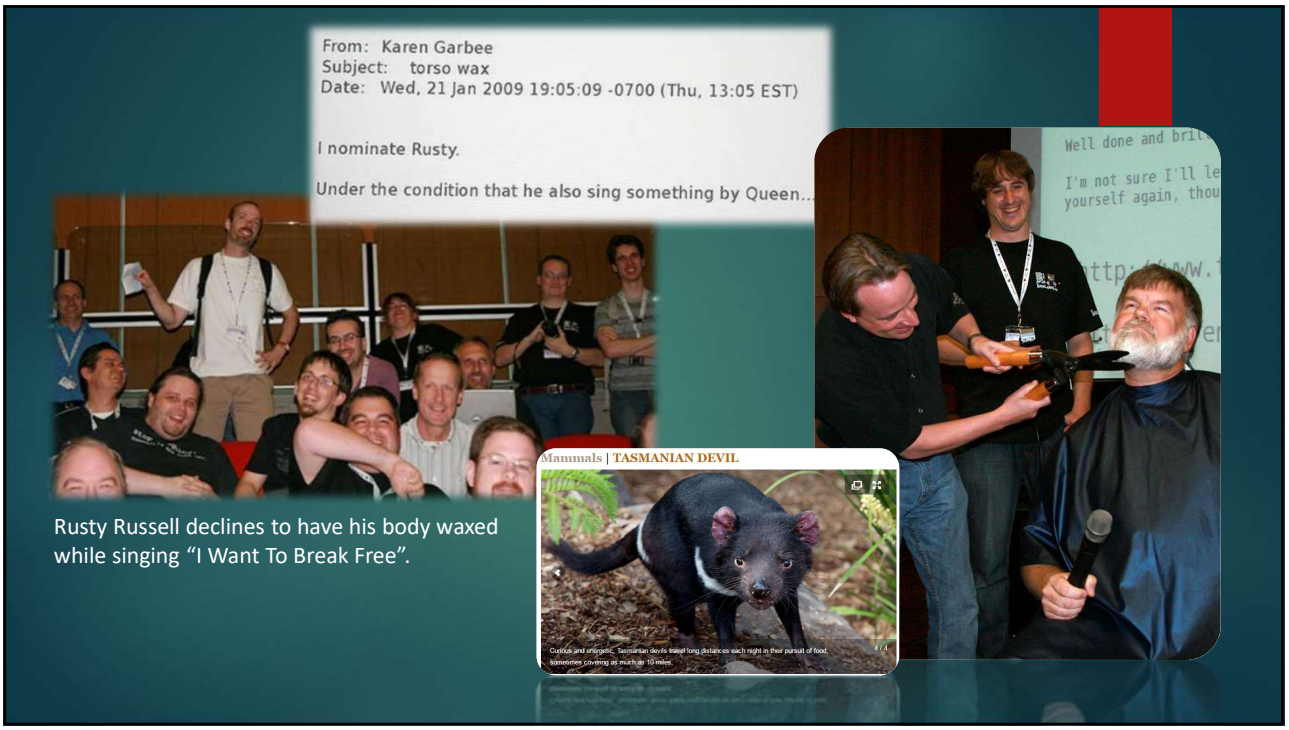
93

测试

```
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i s 200
ioctl cmd s value 200
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i q 0
ioctl cmd q value 200
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i t 500
ioctl cmd t value 500
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i q 0
ioctl cmd q value 500
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i g 0
ioctl cmd g value 500
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i x 1000
ioctl cmd x value 500
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i g 0
ioctl cmd g value 1000
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i h 2000
ioctl cmd h value 1000
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i g 0
ioctl cmd g value 2000
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i r 0
ioctl cmd r value 0
ge@gewubox:~/work/llaolao3$ sudo ./baner 0 i g 0
ioctl cmd g value 800
```

```
[14626.882143] huadeng_open
[14626.882168] huadeng_ioctl cmd 0x40046f01 arg bf8e26fc
[14626.882483] huadeng_release
[14639.400907] huadeng_open
[14639.400923] huadeng_ioctl cmd 0x6f04 arg 0
[14639.401198] huadeng_release
[14646.037893] huadeng_open
[14646.037911] huadeng_ioctl cmd 0x6f02 arg 1f4
[14646.038095] huadeng_release
[14649.080834] huadeng_open
[14649.080844] huadeng_ioctl cmd 0x6f04 arg 0
[14649.081021] huadeng_release
[14659.064501] huadeng_open
[14659.064515] huadeng_ioctl cmd 0x80046f03 arg bf8c6a4c
[14659.069219] huadeng_release
[14667.575551] huadeng_open
[14667.575565] huadeng_ioctl cmd 0xc0046f05 arg bf9b2c1c
[14667.575725] huadeng_release
[14672.928217] huadeng_open
[14672.928229] huadeng_ioctl cmd 0x80046f03 arg bf89a8bc
[14672.928448] huadeng_release
[14680.703269] huadeng_open
[14680.703283] huadeng_ioctl cmd 0x6f06 arg 7d0
[14680.703331] huadeng_release
[14685.496515] huadeng_open
[14685.496579] huadeng_ioctl cmd 0x80046f03 arg bfdd4c9c
[14685.498310] huadeng_release
[14690.295448] huadeng_open
[14690.295456] huadeng_ioctl cmd 0x6f00 arg 0
[14690.295589] huadeng_release
[14692.368731] huadeng_open
[14692.368749] huadeng_ioctl cmd 0x80046f03 arg bfc10f3c
[14692.370297] huadeng_release
```

94



Rusty Russell declines to have his body waxed while singing "I Want To Break Free".

95

$ od - dump files in octal

bash> od –x /dev/random

0000000 7331 9028 7c89 4791 7f64 3deb 86b3 7564

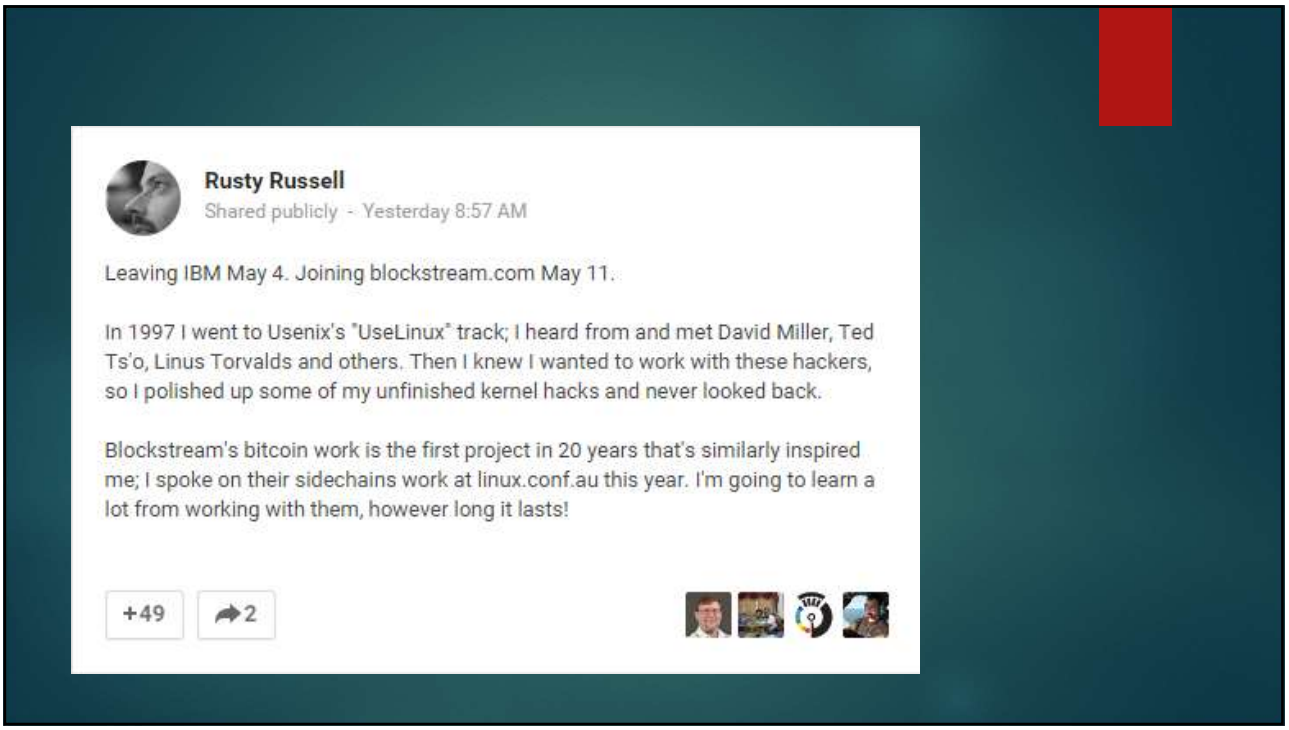
0000020 ebb9 e806 221a b8f9 af12 cb30 9a0e cc28

0000040 68d8 0bbf 68a4 0898 528e 1557 d8b3 57ec

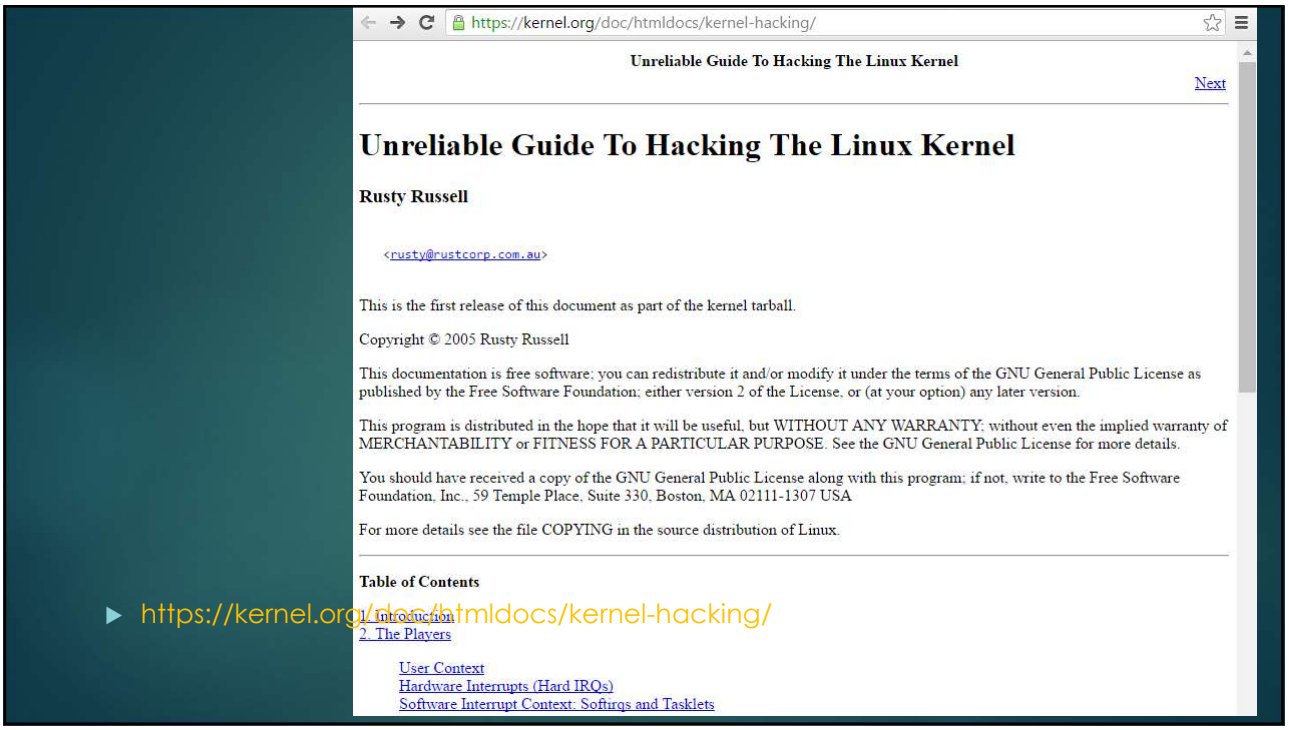
0000060 b01d 8714 b1e1 19b9 0a86 9f60 646c c269

```
ge@gewubox:~/work/cmos$ sudo od -x /dev/cmos1
0000000 0000 0018 ffff ffff ffff 0000 0000 0000
0000020 0000 0000 0000 0000 0000 0000 0000 0000
*
0000200 0000 0018 ffff ffff ffff 0000 0000 0000
0000220 0000 0000 0000 0000 0000 0000 0000 0000
*
0003760 0000 0000 0000 0000
0003770
```

96



97



- https://kernel.org/doc/htmldocs/kernel-hacking/

98

切问而近思

欢迎关注格友公众号