Welcome potential kernel hackers

Christoph Hellwig

hch@lst.de

What's a kernel hacker?

`http://en.wiktionary.org/wiki/Kernel_hacker`:
Kernel hacker

1. A title of developers who maintain the Linux kernel code.

What's a kernel hacker?

`http://en.wiktionary.org/wiki/Kernel_hacker`:
Kernel hacker

1. A title of developers who maintain the Linux kernel code.

- by the way, I'm a kernel hacker

Is a kernel hacker something special?

Is a kernel hacker something special?

YES

- The Kernel runs in privileged mode and can crash the whole system
- Kernel hackers are treated as gods by the free software community

Is a kernel hacker something special?

YES

- The Kernel runs in privileged mode and can crash the whole system
- Kernel hackers are treated as gods by the free software community

NO

- The kernel is just another big, heavily multi-threaded program
- A well written and very modular one

Why I am here

Why I am here

The Linux kernel wants you!

- We need more young programmers from all over the world
- The Linux kernel is growing a lot
- Fresh blood is always a good thing for projects
- Some of us are getting older and we want to keep the quality up

- Argentinia lacks kernel hackers!

Why to become a kernel hacker

- It's easy (at the beginning)
- Allows to grow with the job into more difficult areas
- Kernel hacking includes:
  - Programming hardware
  - Dealing with scaling problems
  - Refactoring large old code bases
  - Implementing advanced data structures
  - Designing widely-used APIs
  - ... and and and ..

Where to start?

Hello World of course!

Hello World

Source code:

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

static int __init hello_init(void)
{
        printk("Me gusta mucho el lomo\n");
        return 0;
}

module_init(hello_init);
```

Makefile:

```
obj-m += hello.o
```

Easy, eh?

- Kernel mode hello world is two more lines than userspace
- A lot easier than Java at least..

Doing a little more

- Just printing to the log is not useful
- Let's implement a real device node

Character Device

```
static ssize_t hello_read(struct file *file, char __user *buf,
                  size_t count, loff_t *ppos)
{
        static const char message[] =
                  "Me gusta mucho la Milanesa\n";
        static ssize_t len = sizeof(message);

        if (*ppos != 0)
                  return 0;

        if (copy_to_user(buf, message, len))
                  return -EFAULT;
        *ppos += len;
        return len;
}
```

Character Device (cont.)

```
struct file_operations hello_fops = {
        .open           = nonseekable_open,
        .read           = hello_read,
};

struct miscdevice hello_misc = {
        .minor          = MISC_DYNAMIC_MINOR,
        .name           = "hello",
        .fops           = &hello_fops,
};

static int __init hello_init(void)
{
        return misc_register(&hello_misc);
}

module_init(hello_init);
```

More advanced kernel hacking

Just reading a string from a device is boring. Let's do something fancy:

More advanced kernel hacking

Just reading a string from a device is boring. Let's do something fancy: Play with real hardware.

Cheap and fast "real" hardware

- Real hardware is expensive to design
- But qemu can fake real hardware
- Let’s create a trivial PCI device

PCI probing

```
static int __devinit hello_probe(struct pci_dev *pdev,
                const struct pci_device_id *id)
{
        int error;

        error = pci_enable_device(pdev);
        if (error)
                return error;
        error = pci_request_regions(pdev, "hello");
        if (error)
                goto out_disable;
        hello_print_signature(pdev);
        return 0;
 out_disable:
        pci_disable_device(pdev);
        return error;
}
```

PCI probing (cont.)

```
static struct pci_device_id hello_pci_tbl[] = {
        { PCI_DEVICE(0x100c, 0x6666), 0, 0, 0},
        { 0, }
};
MODULE_DEVICE_TABLE(pci, hello_pci_tbl);

static struct pci_driver hello_driver = {
        .name           = "hello",
        .id_table       = hello_pci_tbl,
        .probe          = hello_probe,
};

static int __init hello_init(void)
{
        return pci_register_driver(&hello_driver);
}

module_init(hello_init);
```

PCI probing (cont.)

```
#define HELLO_SIG_LEN 5

static void hello_print_signature(struct pci_dev *pdev)
{
        unsigned long base;
        int i;

        printk(KERN_INFO "Found hello device.  Signature: ");

        base = pci_resource_start(pdev, 0);
        for (i = 0; i < HELLO_SIG_LEN; i++)
                printk("%c", inb(base + i));
        printk("\n");
}
```

Demo

Kernelnewbies.org

The website `http://www.kernelnewbies.org` has all resources needed for a new kernel hacker:

- Documentation
- Glossaries to understand Kernelspeak
- Mailinglist for Newbie questions
- Regional Kernelnewbies communities, for example
  - Spanish language community
  - A large Brazilian community

Closing Note

This presentation and the examples will be available on my website, `http://verein.lst.de/~hch/`.

Questions?