HelloDriverModule

原料：

iTOP4412 开发板、U 盘、PC 机

虚拟机 Vmware、Ubuntu12.04.2

源码文件夹“HelloDriverModule”

预备课程：

Linux 无界面最小系统

Linux 内核的编译

Linux 常用命令

学习目标：

了解驱动最简结构

模块化编译驱动

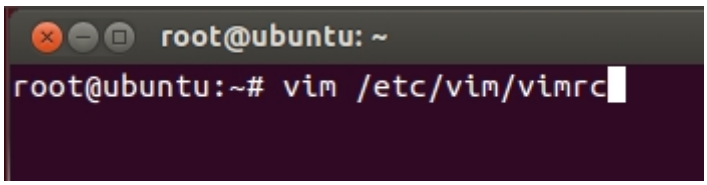
加载驱动、查看驱动、卸载驱动

小贴士之 Vim 显示汉化

如果大家在 Ubuntu 中使用 Vim 编辑器查看源码，中文字符会显示为乱码。

下面给大家介绍一下如何正常显示中文。

打开 vim 的环境变量文件“/etc/vim/vimrc”，使用命令：vim /etc/vim/vimrc，如下图所示。



然后在文件“/etc/vim/vimrc”最底端加入以下代码：

```
set fencs=utf-8,GB18030,ucs-bom,default,latin1
```

如下图所示：

```
if filereadable("/etc/vim/vimrc.local")
  source /etc/vim/vimrc.local
endif

set fencs=utf-8,GB18030,ucs-bom,default,latin1
```

下面截图的为本节代码。

```
#include <linux/init.h>
/*包含初始化宏定义的头文件,代码中的module_init和module_exit在此文件中*/

#include <linux/module.h>
/*包含初始化加载模块的头文件,代码中的MODULE_LICENSE在此头文件中*/


MODULE_LICENSE("Dual BSD/GPL");
/*声明是开源的, 没有内核版本限制*/
MODULE_AUTHOR("iTOPEET_dz");
/*声明作者*/


static int hello_init(void)
{
        printk(KERN_EMERG "Hello World enter!\n");
        /*打印信息, KERN_EMERG表示紧急信息*/
        return 0;
```

源码以及源码分析

itop4412_hello.c 中代码如下：

```
#include <linux/init.h>
/*包含初始化宏定义的头文件，代码中的module_init和module_exit在此文件中*/
#include <linux/module.h>
/*包含初始化加载模块的头文件，代码中的MODULE_LICENSE在此头文件中*/
```

头文件

```
MODULE_LICENSE("Dual BSD/GPL");
/*声明是开源的，没有内核版本限制*/
MODULE_AUTHOR("iTOPEET_dz");
/*声明作者*/
```

声明模块信息

```
static int hello_init(void)
{
    printk(KERN_EMERG "Hello World enter!\n");
    /*打印信息，KERN_EMERG表示紧急信息*/
    return 0;
}

static void hello_exit(void)
{
    printk(KERN_EMERG "Hello world exit!\n");
}
```

功能区

```
module_init(hello_init);
/*初始化函数*/
module_exit(hello_exit);
/*卸载函数*/
```

模块的入口

模块的出口

代码分析：

第一部分：必须包含的头文件 linux/init.h 和 linux/module.h

第二部分：申明区。

在所有的声明中，

```
MODULE_LICENSE("Dual BSD/GPL");
/*声明是开源的，没有内核版本限制*/
```

这一句是最重要的，所有的 Linux 代码必须遵循 GPL 协议，如果不知道 Linux 的 GPL 协议，去查一下资料吧。如果你不声明 GPL 协议，你的模块是无法在 Linux 中使用的！

```
MODULE_AUTHOR("iTOPEET_dz");
/*声明作者*/
```

作者的声明，这个代码是谁写的，这个申明不是必须的。

第三部分：功能区代码。

这里特别提醒只接触过单片机用户或者那些没有接触过操作系统的用户，有没有发现没有 main 函数。Linux 操作系统相当于”一个球 “，我们要做的事情就是在这个球上添加驱动，不用去管这个球是从哪里开始旋转，转到什么地方了。如果你非要找一个 main 函数才罢休，那么你可以吧模块的初始化函数当做 main 函数，也是模块的入口。

在功能区里面，加载驱动和卸载驱动时候调用的函数，这两个函数都只是调用了 printk 函数。

打印函数 printk 想超级终端传递数据，KERN_EMERG 是紧急情况的标识，加载和卸载驱动模块的时候，我们分别打印输出“Hello World enter!” “Hello world exit!\n” 。

打印函数 printk 的 8 个级别如下：

#define KERN_EMERG 0/*紧急事件消息，系统崩溃之前提示，表示系统不可用*/

#define KERN_ALERT 1/*报告消息，表示必须立即采取措施*/

#define KERN_CRIT 2/*临界条件，通常涉及严重的硬件或软件操作失败*/

#define KERN_ERR 3/*错误条件，驱动程序常用 KERN_ERR 来报告硬件的错误*/

#define KERN_WARNING 4/*警告条件，对可能出现问题的情况进行警告*/

#define KERN_NOTICE 5/*正常但又重要的条件，用于提醒*/

#define KERN_INFO 6/*提示信息，如驱动程序启动时，打印硬件信息*/

#define KERN_DEBUG 7/*调试级别的消息*/

第四部分：模块的入口和模块的出口

加载模块和卸载模块。

加载的时候调用了功能区的函数 static int hello_init(void)

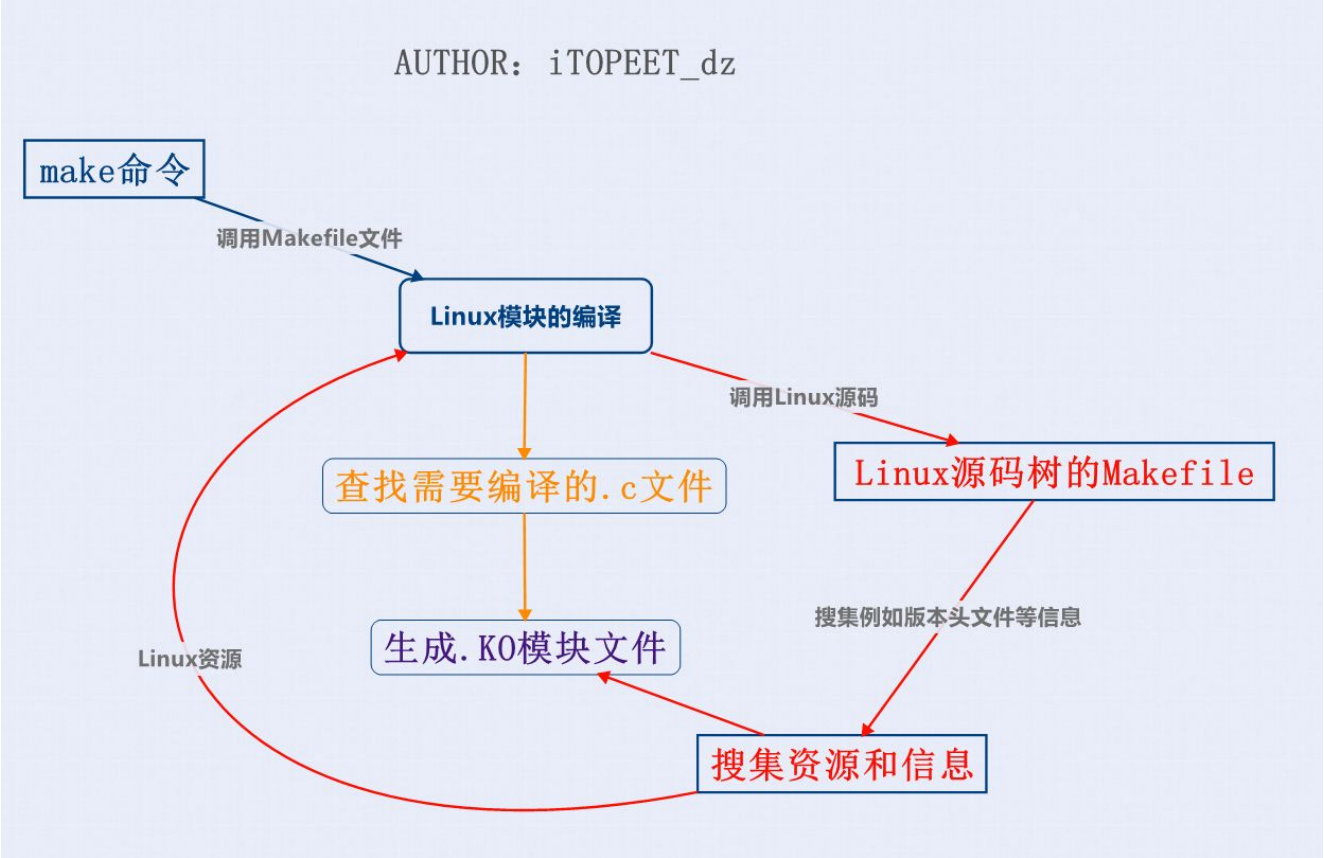
卸载的时候调用了功能区的函数 static void hello_exit(void)

预想的现象：

加载驱动的时候打印：Hello World enter!

卸载驱动的时候打印：Hello world exit!

模块的编译流程图



通过上图可以看到编译中

执行 make 命令之后，调用 Makefile 文件进行 Linux 模块的编译。

Linux 模块的编译分为两个条线。

第一条红色的线：进入 Linux 源码中，调用版本信息以及一些头文件等。

这一条线经过的是整个 Linux 的源码文件。

第二条橙色的线：搜集完 Linux 源码树的信息之后，Makefile 继续执行，调用编译.KO 文件的源码文件，这里是 itop4412_hello.c 这个“iTop4412_Kernel_3.0”整个源码。

这一条线走的是 itop4412_hello.c，虽然都是源码，但是此源码非彼源码。

Makefile 文件通过执行上面两条线，通过搜集到信息，编译生成.KO 模块

编译文件 Makefile

在单片机或者上位机编程的时候，都有集成开发工具。程序员按照开发工具的规则，将代码放入指定的位置，通常是一个 main.c 文件加上很多.c 文件，代码写好了，开发工具中某个按钮一点，就给你自动编译成了二进制文件。

在 Linux 中，并没有这样的集成开发工具，这里还需要自己写编译文件。

编译文件一般是用脚本编写，脚本成千上万，脚本语言学也学不完，脚本的使用只能说是用到哪里学到哪里。

Makefile 编译文件如下：

```
#通知编译器我们要编译模块的哪些源码
#这里是编译itop4412_hello.c这个文件编译成中间文件itop4412_hello.o
obj-m += itop4412_hello.o

#源码目录变量，这里用户需要根据实际情况选择路径
#作者是将Linux的源码拷贝到目录/home/topeet/android4.0下并解压的
KDIR := /home/topeet/android4.0/iTop4412_Kernel_3.0

#当前目录变量
PWD ?= $(shell pwd)

#make命名默认寻找第一个目标
#make -C就是指调用执行的路径
#$(KDIR)Linux源码目录，作者这里指的是/home/topeet/android4.0/iTop4412_Kernel_3.0
#$(PWD)当前目录变量
#modules要执行的操作
all:
	make -C $(KDIR) M=$(PWD) modules

#make clean执行的操作是删除后缀为o的文件
clean:
	rm -rf *.o
```

具体操作

下面具体介绍如何操作。

Linux 源码文件夹如下图。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls
arch                    Documentation   lib              REPORTING-BUGS
binary                  drivers         MAINTAINERS      samples
block                   firmware        Makefile         scripts
config_for_android      fs              mm               security
config_for_android_2M   include         modem.patch      sound
config_for_linux        init            modules.builtin  System.map
config_for_ubuntu       ipc             modules.order    tools
config_for_ubuntu_hdmi  Kbuild          Module.symvers   usr
COPYING                 Kconfig         net              virt
CREDITS                 kernel          pull_log.bat     vmlinux
crypto                  kernel_readme.txt README         vmlinux.o
```

itop4412_hello.c 文件和 Makefile 文件在如下图所示目录。

```
root@ubuntu:/home/topeet/module/hello# ls
itop4412_hello.c  Makefile
root@ubuntu:/home/topeet/module/hello#
```

在目录“/home/topeet/module/hello 下”，执行 make 命令，生成了 itop4412_hello.ko 文件，即是我们需要的驱动模块文件。如下图所示。

```
root@ubuntu:/home/topeet/module/hello# make
make -C /home/topeet/android4.0/iTop4412_Kernel_3.0 M=/home/topeet/module/hello
modules
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
  CC [M]  /home/topeet/module/hello/itop4412_hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/topeet/module/hello/itop4412_hello.mod.o
  LD [M]  /home/topeet/module/hello/itop4412_hello.ko
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
root@ubuntu:/home/topeet/module/hello# ls
itop4412_hello.c   itop4412_hello.mod.c  itop4412_hello.o  modules.order
itop4412_hello.ko  itop4412_hello.mod.o  Makefile          Module.symvers
root@ubuntu:/home/topeet/module/hello#
```

在开发板中烧写光盘目录\image\linux 下的 zImage 内核文件和 ramdisk-uboot.img 文件。

系统文件烧写最小文件系统，可以在本节实验代码包中找到。

烧写上面三个文件，重启开发板后，文件系统启动后，输入“回车”超级终端如下图。需要注意的是，这个最小系统并不带有显卡驱动，所以显示屏上只有迅为电子的 log。

```
[    6.036709] CPU1: shutdown
[    6.044306] link_reset() speed: 10 duplex: 0
[    6.050125] ADDRCONF(NETDEV_UP): eth0: link is not ready
Done

Please press Enter to activate this console. [    6.979016]

[root@iTOP-4412]#
```

然后我们将 itop4412_hello.ko 驱动文件拷贝到 U 盘或者 TF，然后接到开发板上。

将 U 盘或者 TF 卡加载，可以参考使用手册 11.3.3 qt 挂载盘符。

我这里使用的是 U 盘，挂载之后我们需要的加载模块驱动 KO 文件，在 mnt/udisk 下。如下图所示。

```
[root@iTOP-4412]# mount /dev/sda1 /mnt/udisk/
[root@iTOP-4412]# ls /mnt/udisk/
itop4412_hello.ko
[root@iTOP-4412]#
```

然后使用命令： insmod /mnt/udisk/itop4412_hello.ko 加载模块，如下图所示。

```
[root@iTOP-4412]# insmod /mnt/udisk/itop4412_hello.ko
[  610.394945] Hello World enter!
[root@iTOP-4412]#
```

如上图所示，可以看到我们加载过程中打印了信息“ Hello World enter!”，说明驱动已经加载到 Linux 中去了。

使用命令：lsmod，查看我们模块的信息。

```
[root@iTOP-4412]# lsmod
itop4412_hello 700 0 - Live 0xbf000000
[root@iTOP-4412]#
```

我们使用命令：rmmod itop4412_hello 卸载驱动模块。如下图所示，会报错无法卸载，这里我们新建提示没有的文件夹 ，使用命令：mkdir lib/modules/3.0.15。

```
[root@iTOP-4412]# rmmod itop4412_hello
rmmod: can't change directory to '3.0.15': No such file
[root@iTOP-4412]# mkdir lib/modules/3.0.15
[root@iTOP-4412]#
[root@iTOP-4412]#
```

我们再使用卸载驱动模块的命令：mmod itop4412_hello，如下图所示，可以看到打印出了我们在卸载驱动函数里面添加的打印命令：Hello world exit！

最后使用命令：lsmod，对比前面的 lsmod，发现已经没有了加载的模块驱动了。

```
[root@iTOP-4412]# rmmod itop4412_hello
[  342.064511] Hello world exit!
[root@iTOP-4412]# lsmod
[root@iTOP-4412]#
```