Linux/Android开发记录 学习、记录、分享Linux/Android开发技术

个人资料

liuhaoyutz

访问： 80602次
积分： 1673分
排名： 第7877名

原创： 83篇 转载： 0篇
译文： 0篇 评论： 59条

LDD3源码分析之hello.c与Makefile模板

分类： LDD3源码分析 2012-03-22 14:28 2702人阅读 评论(4) 收藏 举报

makefile linux内核 linux module 工作 编程

作者：刘昊昱

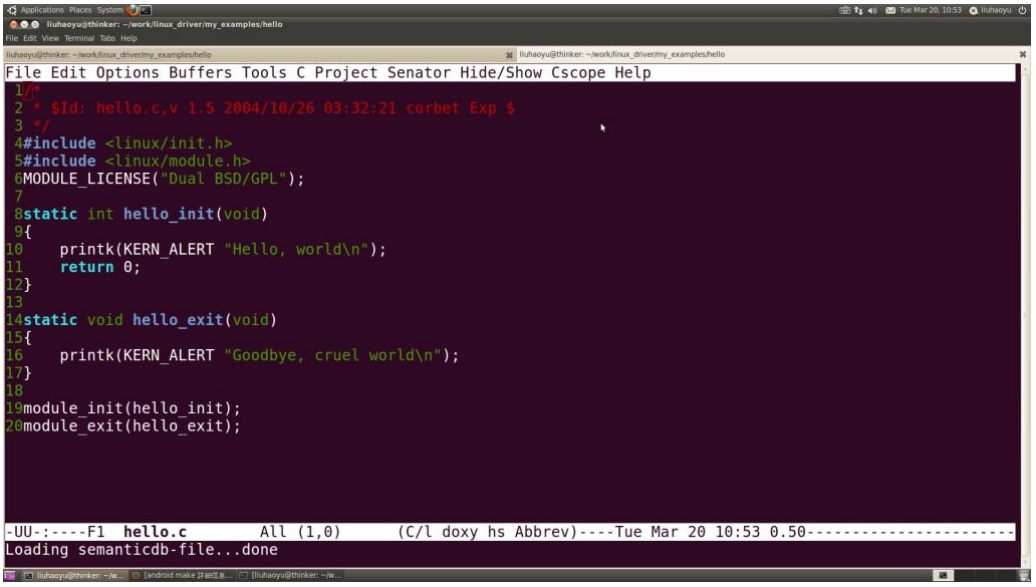
博客：http://blog.csdn.net/liuhaoyutz

编译环境：Ubuntu 10.10

内核版本：2.6.32-38-generic-pae

LDD3源码路径：examples/misc-modules/hello.c

一、hello.c文件分析



这个程序非常简单，它的目的是向我们展示Linux模块编程的架构，而Linux设备驱动程序的开发方法，就是利用了Linux模块编程。

首先来分析一下这个程序。对于任何一个模块程序，不论是简单如这个hello.c，还是复杂如usb模块的代码，我们要分析其源码，首先要找的是module_init和module_exit两个宏。module_init宏的参数是模块被装载时要调用的函数，这是我们分析的起点。而module_exit宏的参数，则是模块被卸载时要调用的函数，这是完成最后清理工作的地方。所以对于hello.c，编译成模块后，装载模块时，hello_init函数就会被调用，打印了一句话“Hello,world”，卸载模块时，hello_exit函数就会执行，打印“Goodbye, cruel world”。

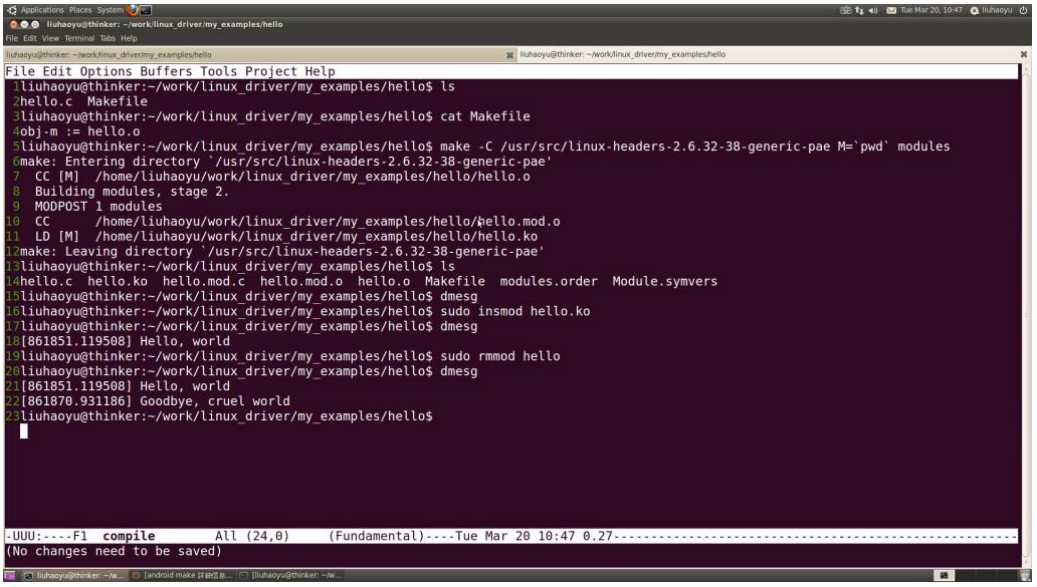
二、最简单的Makefile

要把hello.c编译成内核模块，需要配置好内核源码树，还需要我们提供一个Makefile文件。下面我们来看Makefile的写法。

最简单的Makefile只需要一句话：

obj-m := hello.o

我们先来看一下整个编译过程，再解释这一句话是什么意思。如下图所示：



首先解释一下上图中各命令的作用：

第1行，执行ls命令，可以看到当前目录下有hello.c和Makefile两个文件。

第3行，执行cat Makefile命令，可以看到Makefile的内容只有一句话obj-m := hello.o

第5行，执行make -C /usr/src/linux-headers-2.6.32-38-generic-pae M=`pwd` modules命令，编译hello模块。注意，-C后跟的是内核源码树所在目录(根据自己的配置指定相应路径)。M=后面是反引号(在ESC按键下面)而不是单引号，表示把pwd命令执行的结果(即当前路径)赋值给M。

第13行，执行ls命令，显示编译后当前目录下的内容，可以看到，已经生成了hello.ko，即hello模块。

第15行，执行dmesg命令，可以看到，没有任何信息输出。

第16行，执行sudo insmod hello.ko命令，安装hello模块。

第17行，再次执行dmesg命令，可以看到”Hello, world”信息，这就是刚才安装hello模块时，模块初始化函数hello_init函数打印的语句。

第19行，执行sudo rmmod hello命令，从内核中删除hello模块。注意，指定模块名时用的是hello，而不是hello.ko。

第20行，再次执行dmesg命令，可以看到，除了刚才安装模块时hello_init打印的”Hello, world”，又多了一条语句”Goodbye, cruel world”，这句话就是模块卸载函数hello_exit函数打印的。

至此，可以看到hello模块的编译，安装，卸载都成功了。

上面介绍的这个最简单的Makefile只有一句话” obj-m := hello.o”，显然，按照标准的Makefile语法，这个Makefile应该无法完成任何编译工作才对，但是从实际编译过程可以看出，通过这个Makefile确实把hello模块编译出来了。它是怎么做到的呢？正如LDD3上所说的“问题的答案当然是内核构造系统处理了其余的问题”。也就是说，我们要把这个Makefile放在Linux内核编译系统这个大环境下使用，这样才能编译出hello模块。如果没有Linux内核编译系统，只有这一个Makefile文件，肯定是无法完成编译工作的。

那么我们怎么把这个Makefile放在Linux内核编译系统这个大环境下来使用呢？答案就是上图第5行执行的命令make -C /usr/src/linux-headers-2.6.32-38-generic-pae M=`pwd` modules。这里make命令的”-C”选

2014年01月 (1)
2013年12月 (6)
展开

文章搜索

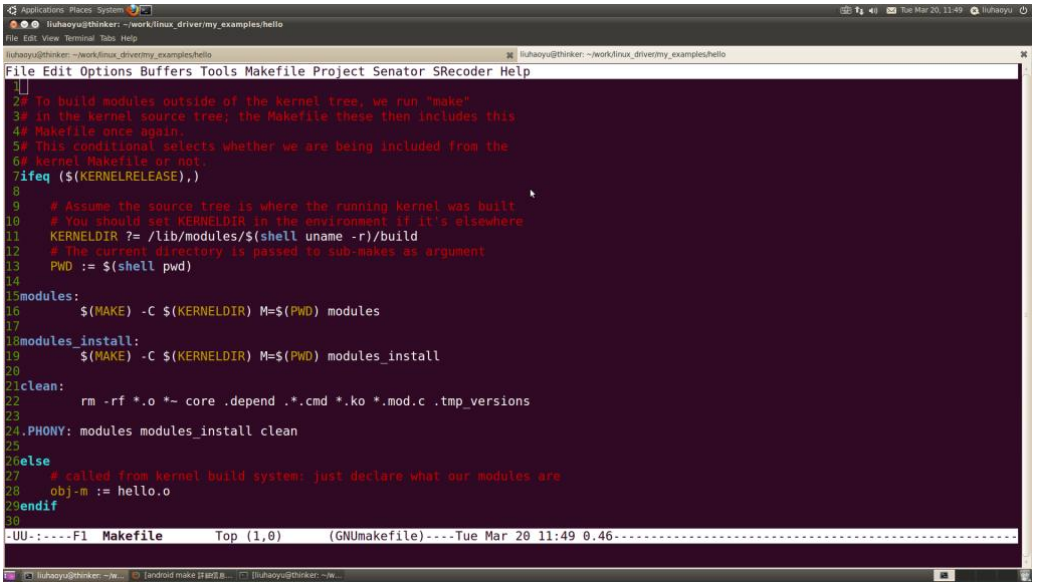
推荐文章

项，指定了内核源码树所在的路径，其中保存了Linux内核源码顶层Makefile，这个顶层Makefile即Linux内核编译系统的入口点。”M=”选项，表明在构造modules目标之前，返回到当前目录，即把要生成的modules目标放在当前目录下。

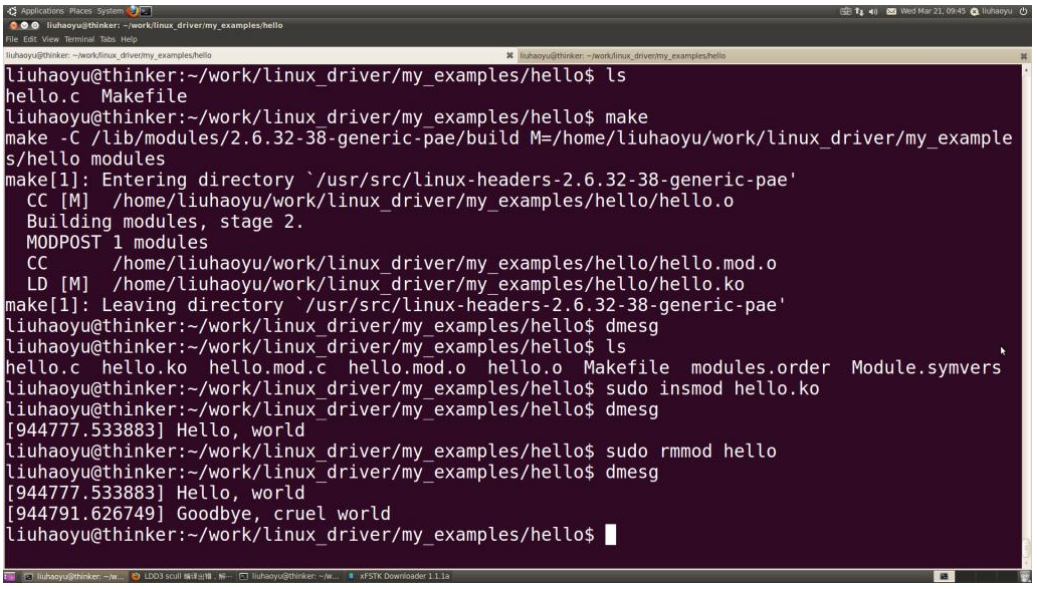
现在可以来看我们的Makefile中的这一句话了” obj-m := hello.o”，这句话表明，当执行make modules命令时(这里忽略”-C”和”M=”选项)，要求从hello.o文件来生成一个目标模块，该目标模块名为hello.ko。

三、功能完整的Makefile

我们前面用的Makefile很简单，功能比较单一，LDD3为我们提供了一个比较好的Makefile模板，简单修改即可拿来编译自己的模块，如下图所示：



使用这个Makefile，编译hello模块的过程显示如下：



下面我们分析一下这个加强版的Makefile的内容。注意，这里需要你对Makefile的基本语法有一定了解，如果对Makefile的基本语法不了解，请先学习相关知识。

第7行，ifeq ($(KERNELRELEASE),)，判断KERNELRELEASE变量是否为空，如果为空则继续向下到11行执行。如果不为空，即已经定义了KERNELRELEASE，说明是从内核编译系统调用的，则跳到第26行执行。其效果就和我们前面只有一句话的最简单的Makefile相同了。

第11行，KERNELDIR ?= /lib/modules/$(shell uname -r)/build，给KERNELDIR变量赋值，该变量保存内核源码树所在的路径。注意，linux各发行版本会把内核源码树的一个符号链接放在/lib/modules/$(shell uname -r)/build，对于我的系统，这个符号链接指向的实际就是/usr/src/linux-headers-2.6.32-38-generic-

pae。

第13行，PWD := $(shell pwd)，给PWD变量赋值，该变量保存当前路径。

第16行，$(MAKE) -C $(KERNELDIR) M=$(PWD) modules，如果在命令行执行make modules命令，则相应会执行这条命令编译模块。

第19行，$(MAKE) -C $(KERNELDIR) M=$(PWD) modules_install，如果在命令行执行make modules install命令，则相应会执行这条命令安装模块。

第22行，rm -rf *.o *~ core .depend .*.cmd *.ko *.mod.c .tmp_versions，如果在命令行执行make clean命令，则相应会执行这条命令删除指定文件。

第28行，如果第7行的判断不成立，则会执行这条指令。与前面我们介绍的最简单的Makefile效果相同。

这时我来提出一个问题：

按照上面对Makefile的分析，当我们在命令行执行make命令时，因为KERNELRELEASE变量为空，所以会执行给变量KERNELDIR和PWD的赋值，然后执行第一个目标modules对应的命令，即第16行$(MAKE) -C $(KERNELDIR) M=$(PWD) modules，编译模块。但现在并没有指定obj-m := hello.o，按照我们对第一个最简单的Makefile的分析，这个变量obj-m := hello.o是非常重要的，它指定了当执行make modules命令时(这里忽略”-C”和”M=”选项)，要求从hello.o文件来生成一个目标模块，该目标模块名为hello.ko。因为没有指定obj-m := hello.o，那么编译系统怎么知道要编译的目标模块是什么，怎么编译这个目标模块呢？

答案在LDD3上也有列出了，大家可以理解一下LDD3（中文版）第30页上的第一段文字。概括一下：我们在命令行执行make时，这个makefile文件将会被调用两次，第一次调用时，因为没有设置KERNELRELEASE，所以会设置KERNELDIR和PWD变量，然后执行modules目标对应的命令，即第16行$(MAKE) -C $(KERNELDIR) M=$(PWD) modules。执行这个命令时，”-C”选项决定了首先会创建内核构造系统(其中包括创建KERNELRELEASE变量)，然后再按照该makefile执行，这次执行，因为已经定义了KERNELRELEASE变量，所以会直接跳到26行对应的else语句执行，并在第28行设置了obj-m := hello.o。后面的过程就和我们前面介绍的最简单的只有一句话的Makefile一样了。

四、总结

本文首先通过分析LDD3自带的hello.c程序，介绍了Linux模块编程的概念，Linux设备驱动程序就是建立在Linux模块编程的基础上。本文的另一个重点是介绍了Linux设备驱动程序makefile的写法及工作原理，以后我们的驱动程序，就可以使用这里介绍的makefile模板来完成编译工作。

更多 0

下一篇 LDD3源码分析之字符设备驱动程序

顶 9 踩 0

主题推荐 源码 makefile 工作 编程 ubuntu

猜你在找

《Linux设备设备驱动程序（第三版）》学习笔记之一：写自己的函数直接调用Linux system call
web服务器thttpd的移植
STM32 对内部FLASH读写接口函数
find 与 grep
Linux字符设备驱动(二)
I2C总线串行串行输入输出结构
android图形系统详解四：控制硬加速
linux 修改默认语言环境
Hello world的Makefile写法

查看评论

4楼 fantasyhujian 2013-10-15 17:23发表

写的很详细，对初学者很有帮助！！！

3楼 雁子依然 2013-07-02 14:45发表

今天开始拜读你的这个系列文章，有个小问题，关于KERNELRELEASE，我的理解是不是也可以不用去判断这个变量的定义呢？

我试着把ifeq else endif这三行注释，也可以编译成功，我理解是不是这样：

1. 在模块所在目录执行make，根据makefile规则，会先去找第一个目标modules，执行后面的命令$(MAKE) -C $(KERNELDIR) M=$(PWD) modules

2. 进入kernel目录，kbuild开始工作

3. 根据M=参数，生成模块之前回到当前目录再次读取makefile里面的obj-m

所以是不是说KERNELRELEASE定义与否，效果都是一样的呢

当然有一个例外就是，如果在当前目录下make的时候直接就make KERNELRELEASE=y，那就会编译出错，但是应该不会有人这么做吧，或者说不会出现这样的情况吧？

可以帮的解答一下么？

2楼 moyang0501 2012-04-13 14:33发表

顶一下

哈哈

1楼 cibon 2012-03-23 10:32发表

写的很详细啊，尤其是Makefile部分：-）