**实 验 报 告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **组 别** |  | **姓 名** | 尹浩男 | **同组实验者** |  |
| **实验项目名称** | 添加简单的Linux内核模块 | | | **实验日期** | 10月26日 |
| **教师评语:** |  | | | | |
| **实验成绩：** | | | **指导教师（签名）：**  2020年 月 日 | | |
| **一．实验目的**  1. 掌握基本的Linux内核模块开发框架和编译方法  2. 掌握Linux内核模块添加流程  3. 理解Linux内核模块代码中的一些常见宏和参数  4. 掌握Linux内核模块程序和应用程序的差异  **二．实验内容**  1. 添加最简单的内核模块kello  2. 完成kello内核模块的添加和撤销实验  3. 阅读和理解内核模块kello源代码及其Makefile  4. 理解、掌握应用程序和内核模块代码的差异、编译和运行的差异  **三. 实验步骤**  3.2demo模块的构建  Demo.c   |  | | --- | | #include <linux/module.h>  int kello\_init( void )  {      printk( "\n   Hello, 尹浩男 \n" );      return  0;  }  void kello\_exit( void )  {      printk( "\n   Goodbye now... 尹浩男 \n" );  }  MODULE\_AUTHOR("尹浩男");  MODULE\_LICENSE("GPL");  module\_init(kello\_init);  module\_exit(kello\_exit); |   Makefile   |  | | --- | | ifneq   ($(KERNELRELEASE),)  obj-m   := kello.o  else  KDIR    := /lib/modules/$(shell uname -r)/build  PWD := $(shell pwd)  default:      $(MAKE) -C $(KDIR) M=$(shell pwd) modules      rm -r -f .tmp\_versions \*.mod.c .\*.cmd \*.o \*.symvers  endif |   **四. 实验结果**  实验执行过程如下图所示   |  | | --- | | > cd /project/kello  > make  # 编译模块代码  make -C /lib/modules/5.4.0-42-generic/build M=/project/kello modules  make[1]: Entering directory '/usr/src/linux-headers-5.4.0-42-generic'    CC [M]  /project/kello/kello.o    Building modules, stage 2.    MODPOST 1 modules    CC [M]  /project/kello/kello.mod.o    LD [M]  /project/kello/kello.ko  make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-42-generic'  rm -r -f .tmp\_versions \*.mod.c .\*.cmd \*.o \*.symvers  > insmod kello.ko   # 安装模块  > dmesg | tail -1   #  显示最后一行                       Hello, 尹浩男  > rmmod kello.ko  > dmesg | tail -1  # 显示最后一行                      Goodbye now... 尹浩男 |   **五. 分析**  **模块的组成如下方介绍**   |  | | --- | | 添加模块  模块的意义  由于LINUX设备驱动以内核模块的形式而存在，在具体的设备驱动开发中，将驱动编译为内核模块也有很强的工程意义，  因为如果将正在开发中的驱动直接编译入内核，而开发过程中会不断修改驱动的代码，  则需要不断地编译内核并重启内核，但是如果编译为模块，则只需要rmmod并insmod即可，开发效率大为提高。  下面说明如何添加、编译并允许LINUX模块。  模块的组成  LINUX的模块主要由6部分组成：  1. 模块的加载函数（必须）  当通过insmod或modprobe命令加载内核模块时，模块的加载函数会自动被内核执行，完成本模块的相关初始化工作。  2. 模块的卸载函数（必须）  当通过rmmod命令卸载某模块时，模块的卸载函数会自动被内核执行，完成与模块加载函数相反的功能。  3. 模块许可证声明  模块许可证（LICENSE）声明描述内核模块的的许可权限，如果不声明LICENSE,模块被加载时，将接到内核被污染的警告。  4. 模块参数（可选）  模块参数是模块被加载的时候可以被传递给它的值，它本身对应模块内部的全局变量。  5. 模块导出符号（可选）  内核模块可以导出符号(symbol,对应于函数或者是变量），这样其他模块就可以使用本模块中的变量或者是函数。  6. 模块作者等信息声明（可选） |   dmesg命令   |  | | --- | | dmesg命令用于显示开机信息。  kernel会将开机信息存储在ring buffer中。您若是开机时来不及查看信息，可利用dmesg来查看。开机信息亦保存在/var/log目录中，名称为dmesg的文件里。  持续跟踪dmesg日志文件  添加模块和删除模块并不会改变/var/log/dmesg文件的内容，可能是系统还没有写进去  >  tail -f /var/log/dmesg | | | | | | |