国嵌

www.enjoylinux.cn

Linux内核开发

谢伟 著

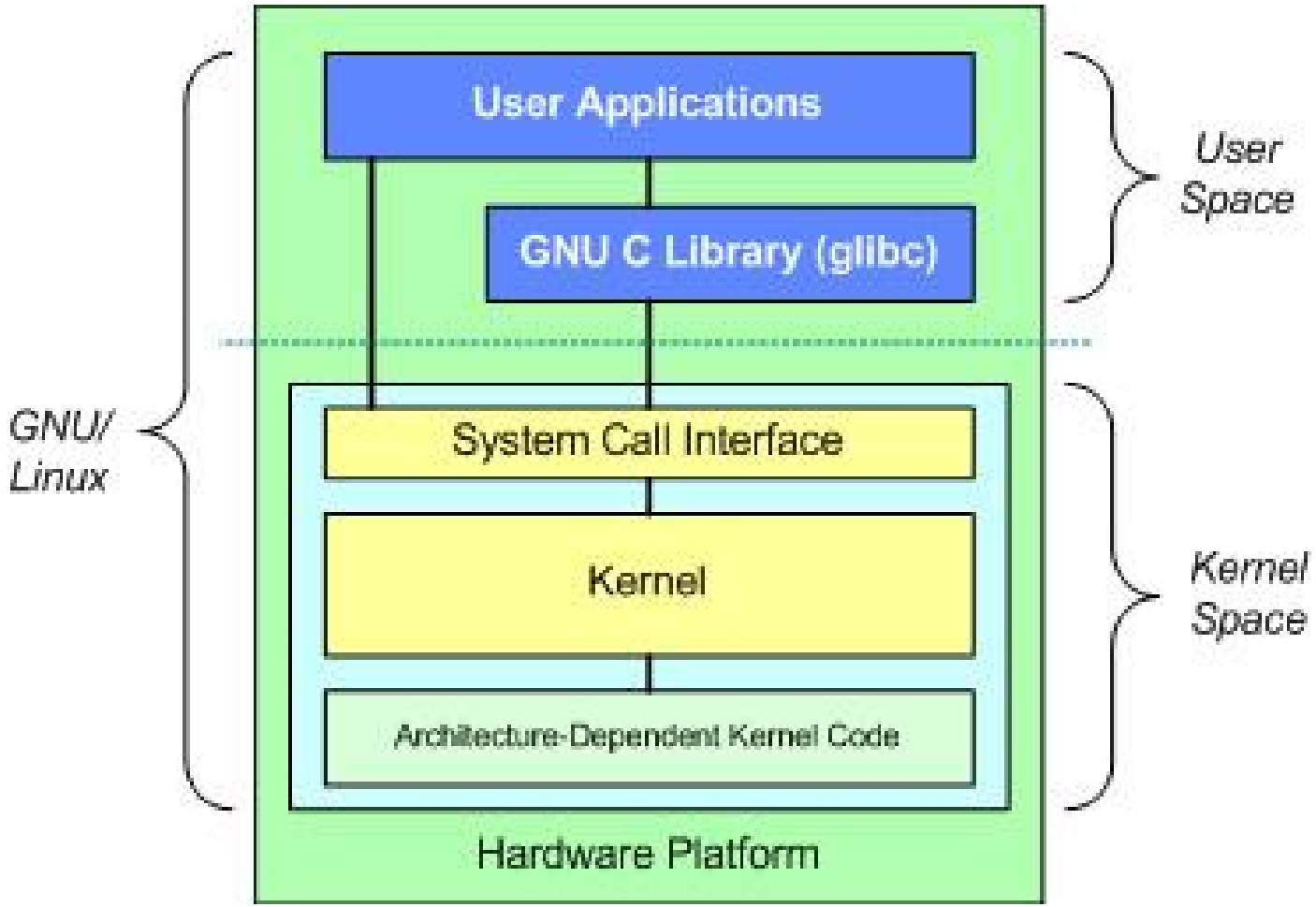
Contents

Contents

Linux体系结构

Linux系统如何构成的？

Linux体系结构


User Applications
GNU C Library (glibc)
System Call Interface
Kernel
Architecture-Dependent Kernel Code
Hardware Platform
GNU/ Linux
User Space
Kernel Space

Linux体系结构

从上图得知，Linux由 用户空间和内核空间 两部分组成。

为什么Linux系统会被划分为用户空间与内核空间？

Linux体系结构

现代CPU通常实现了不同的工作模式，以ARM为例，实现了7种工作模式:

用户模式（usr）、快速中断(fiq)、外部中断(irq)、管理模式（svc）、数据访问中止(abt)、系统模式(sys)、未定义指令异常(und)

Linux体系结构

X86也实现了4个不同的级别：Ring0—Ring3。Ring0下，可以执行特权指令，可以访问IO设备等，在Ring3则有很多限制。

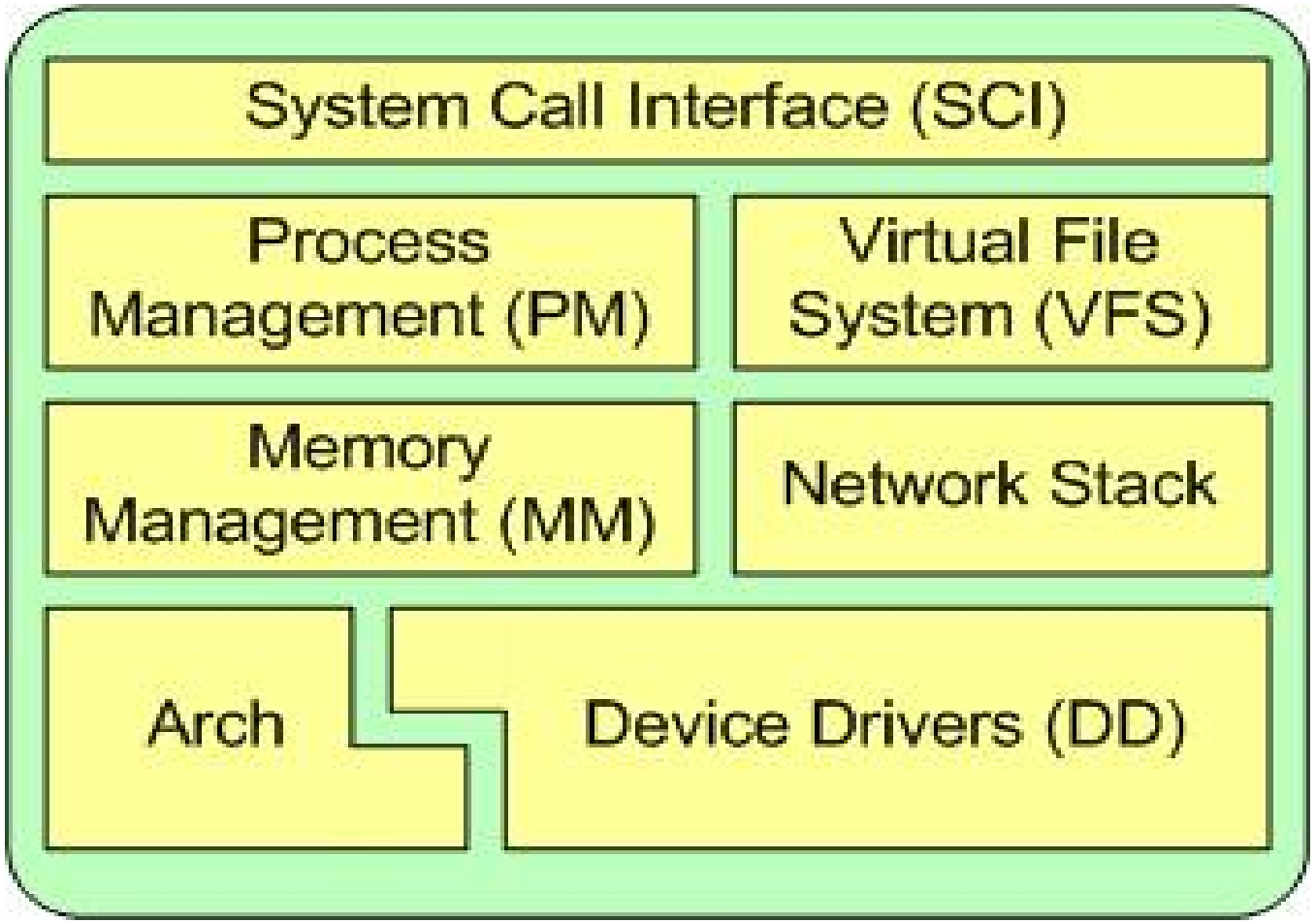
Linux系统利用了CPU的这一特性，使用了其中的两级来分别运行Linux内核与应用程序，这样使操作系统本身得到充分的保护。例如：如果使用X86，用户代码运行在Ring3，内核代码运行在Ring0。

Linux体系结构

内核空间与用户空间是程序执行的两种不同状态，通过系统调用和硬件中断能够完成从用户空间到内核空间的转移。

Linux内核如何构成的？

Linux内核架构


System Call Interface (SCI)
Process Management (PM)
Virtual File System (VFS)
Memory Management (MM)
Network Stack
Arch
Device Drivers (DD)

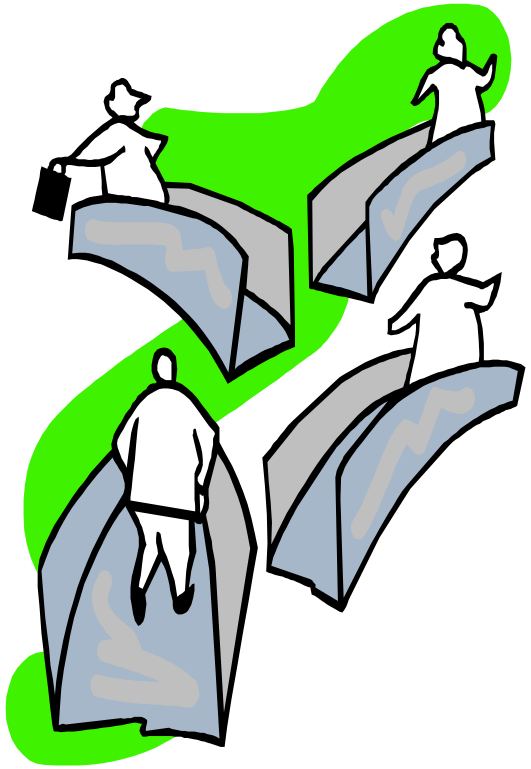
系统调用接口

SCI层为用户空间提供了一套标准的系统调用函数来访问Linux内核，搭起了用户空间到内核空间的桥梁。

进程管理

进程管理的重点是创建进程（fork、exec），停止进程（kill、exit），并控制它们之间的通信（signal 或者 POSIX 机制）。进程管理还包括控制活动进程如何共享CPU，即进程调度。

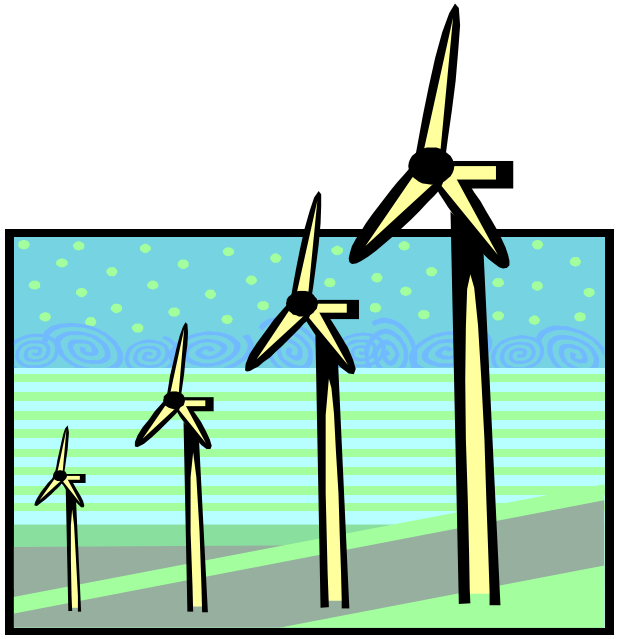
内存管理



内存管理的主要作用是控制多个进程安全地共享内存区域。

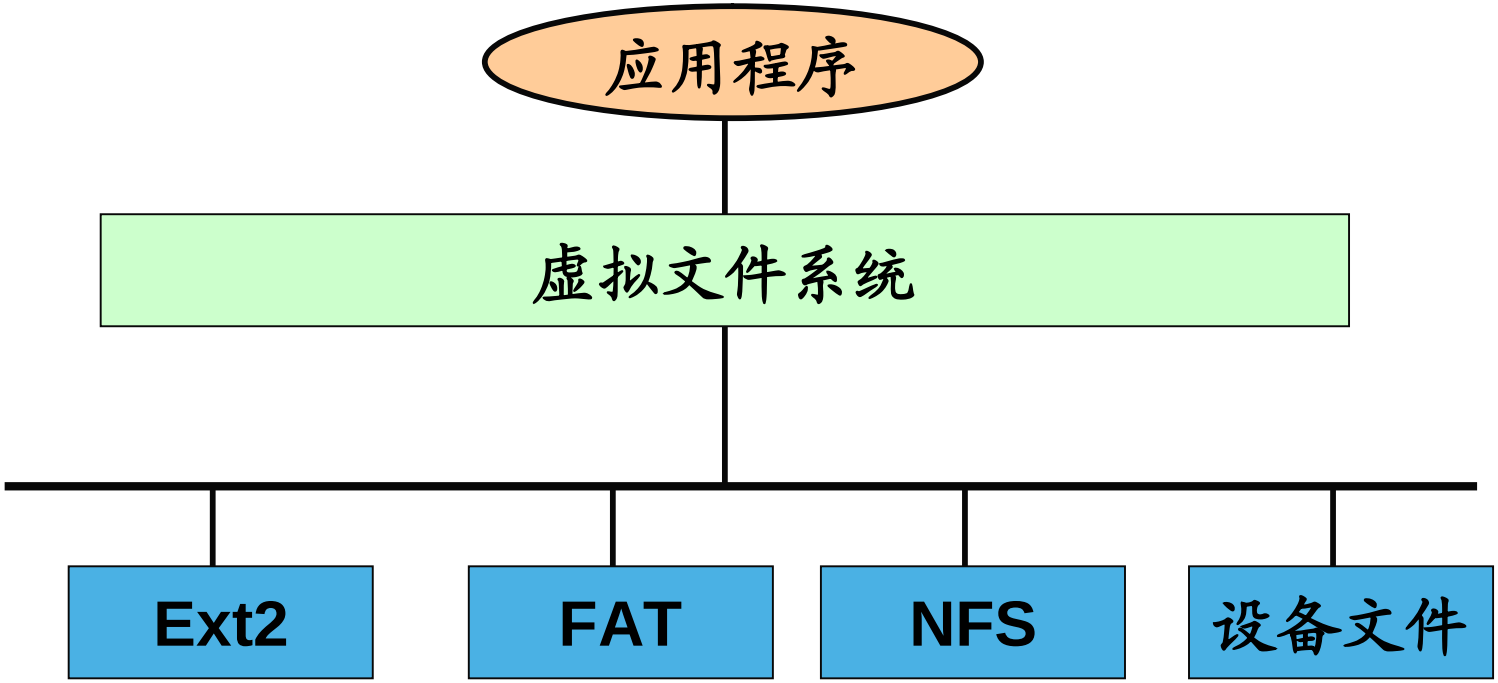
网络协议栈

内核协议栈为Linux提供了丰富的网络协议实现。



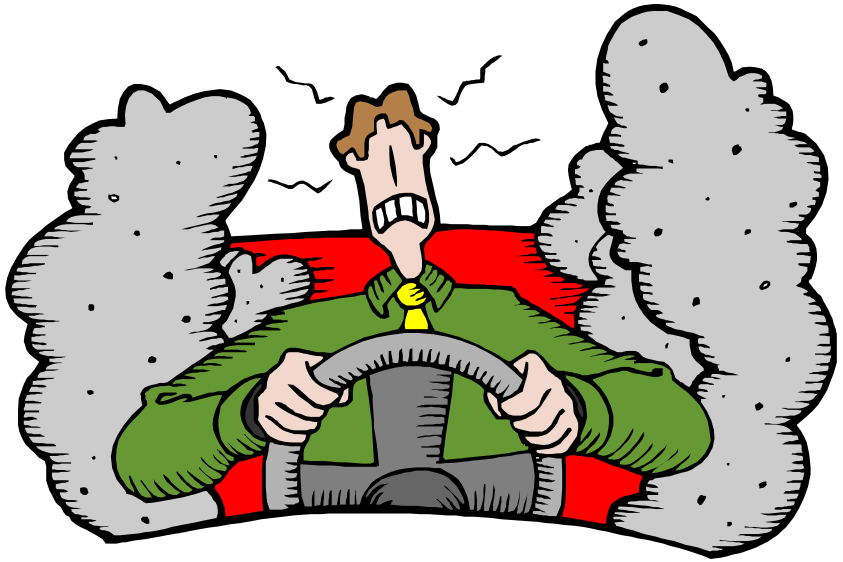
虚拟文件系统（VFS）

VFS隐藏各种文件系统的具体细节，为文件操作提供统一的接口。



设备驱动

Linux 内核中有大量代码都在设备驱动程序中，它们控制特定的硬件设备。



Contents

目录结构

Linux内核源代码采用树形结构进行组织，非常合理地把功能相关的文件都放在同一个子目录下，使得程序更具可读性。

内核源代码下载地址:

www.kernel.org

目录结构


arch
block
crypto
Documentation
drivers
firmware
fs
include
init
ipc
kernel
lib
mm
net
samples
scripts
security
sound
usr
virt

目录结构(展示)

- arch目录

  arch是architecture的缩写。内核所支持的每种CPU体系，在该目录下都有对应的子目录。每个CPU的子目录，又进一步分解为boot,mm,kernel等子目录，分别包含控制系统引导，内存管理，系统调用等。

目录结构

| --x86 /* 英特尔cpu及与之相兼容体系结构的子目录*/

| |--boot /*引导程序*/

| | |--compressed /*内核解压缩*/

| |--tools /*生成压缩内核映像的程序*/

| |--kernel /*相关内核特性实现方式，如信号处理、时钟处理*/

| |--lib /*硬件相关工具函数*/

目录结构

- **block目录**
  部分块设备驱动程序
- **crypto目录**
  加密、压缩、CRC校验算法
- **documentation**
  内核的文档
- **drivers目录**
  设备驱动程序

目录结构

- **fs目录**

  存放各种文件系统的实现代码。每个子目录对应一种文件系统的实现，公用的源程序用于实现虚拟文件系统vfs。

  | |--devpts /* /dev/pts虚拟文件系统*/
  | |--ext2 /*第二扩展文件系统*/
  | |--fat /*MS的fat32文件系统*/
  | |--isofs /*ISO9660光盘cd-rom上的文件系统*/

- **include目录**

  内核所需要的头文件。与平台无关的头文件在include/linux 子目录下，与平台相关的头文件则放在相应的子目录中。

目录结构

- **init目录**

  内核初始化代码

- **ipc目录**

  进程间通信的实现代码

- **kernel目录**

  Linux大多数关键的核心功能都是在这个目录实现。（调度程序，进程控制，模块化）

目录结构

- lib目录

  库文件代码

- mm目录

  mm目录中的文件用于实现内存管理中与体系结构无关的部分（与体系结构相关的部分在哪里实现？）

目录结构

v net目录

网络协议的实现代码

| |--802 /*802无线通讯协议核心支持代码*/

| |--appletalk /*与苹果系统连网的协议*/

| |--ax25 /*AX25无线INTERNET协议*/

| |--bridge /*桥接设备*/

| |--ipv4 /*IP协议族V4版32位寻址模式*/

| |--ipv6 /*IP协议族V6版*/

目录结构

- **samples**
  一些内核编程的范例

- **scripts**
  配置内核的脚本

- **security**
  SElinux的模块

目录结构

- **sound**

  音频设备的驱动程序

- **usr**

  cpio命令实现

- **virt**

  内核虚拟机

Contents

内核配置与编译

Linux内核具有可定制的优点,具体步骤如下:

1. 清除临时文件、中间文件和配置文件.

- **make clean**

  remove most generated files but keep the config

- **make mrproper**

  remove all generated files + config files

- **make distclean**

  mrproper + remove editor backup and patch files

内核配置与编译

2、确定目标系统的软硬件配置情况，比如CPU的类型、网卡的型号，所需支持的网络协议等。

3、使用如下命令之一配置内核：

- **make config**：基于文本模式的交互式配置。
- **make menuconfig**：基于文本模式的菜单型配置。（推荐使用）

内核配置与编译

- **make oldconfig**:

  使用已有的配置文件（.config），但是会询问新增的配置选项。

- **make xconfig**:

  图形化的配置（需安装图形化系统）。

内核配置与编译

make menuconfig 是最为常用的内核配置方式，使用方法如下：

1、使用方向键在各选项间移动；

2、使用“Enter”键进入下一层选单；每个选项上的高亮字母是键盘快捷方式，使用它可以快速地到达想要设置的选单项。

内核配置与编译

3、在括号中按“y”将这个项目编译进内核中，按“m”编译为模块，按“n”为不选择（按空格键也可在编译进内核、编译为模块和不编译三者间进行切换），按“h”将显示这个选项的帮助信息，按“Esc”键将返回到上层选单。

内核配置与编译

配置菜单中的项该怎么选择呢?

内核配置与编译

配置选项说明

内核配置与编译

内核配置通常在一个已有的配置文件基础上，通过修改得到新的配置文件

Linux内核提供了一系列可供参考的内核配置文件，位于

Arch/$cpu/configs

内核配置与编译

4、编译内核：

- make zImage
- make bzImage

区别：在X86平台，zImage只能用于小于512K的内核

*如需获取详细编译信息，可使用：

- make zImage V=1
- make bzImage V=1

** 编译好的内核位于arch/<cpu>/boot/目录下 **

内核配置与编译

5、编译内核模块:

- make modules

6、安装内核模块

- make modules_install

将编译好的内核模块从内核源代码目录copy至/lib/modules下

内核配置与编译

7、制作init ramdisk

mkinitrd initrd-$version $version

例:

mkinitrd initrd-2.6.29 2.6.29

***$version 可以通过查询/lib/modules下的目录得到**

内核安装（X86平台）

1、cp arch/x86/boot/bzImage

/boot/vmlinuz-$version

2、cp $initrd /boot/

3、修改/etc/grub.conf 或者 /etc/lilo.conf

** $version 为所编译的内核版本号**

实验

Linux内核配置与编译

配置、编译、安装

基于PC平台的Linux内核

Contents

功能

什么是内核模块?

Linux内核的整体结构非常庞大，其包含的组件也非常多，如何使用需要的组件呢:

- 方法一：把所有的组件都编译进内核文件，即：zImage或bzImage，但这样会导致两个问题：一是生成的内核文件过大；二是如果要添加或删除某个组件，需要重新编译整个内核。

功能

有没有一种机制能让内核文件(zImage或bzImage)本身并不包含某组件，而是在该组件需要被使用的时候，动态地添加到正在运行的内核中呢？

有，Linux提供了一种叫做“内核模块”的机制，就可以实现以上效果。

特点

内核模块具有如下特点：

- 模块本身并不被编译进内核文件(zImage或者bzImage)
- 可以根据需求，在内核运行期间动态的安装或卸载。

范例(hello world)


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

static int hello_init(void)
{
    printk(KERN_WARNING"Hello, world !\n");
    return 0;
}

static void hello_exit(void)
{
    printk(KERN_INFO "Goodbye,  world\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

程序结构

1、模块加载函数（必需）

安装模块时被系统自动调用的函数，通过module_init宏来指定，在HelloWorld模块中，模块加载函数为？

程序结构

2、模块卸载函数（必需）

卸载模块时被系统自动调用的函数，通过module_exit宏来指定，在HelloWorld模块中，模块卸载函数为？

模块的编译

在Linux 2.6下编译模块，多使用makefile

范例 makefile 分析

范例 多文件makefile 分析

安装与卸载

- 加载 insmod (insmod hello.ko)
- 卸载 rmmod (rmmod hello)
- 查看 lsmod
- 加载 modprobe （modprobe hello）

modprobe 如同 insmod, 也是加载一个模块到内核。它的不同之处在于它会根据文件

/lib/modules/<$version>/modules.dep

来查看要加载的模块, 看它是否还依赖于其他模块，如果是，modprobe 会首先找到这些模块, 把它们先加载到内核。

对比

对比应用程序，内核模块具有以下不同：

应用程序是从头(main)到尾执行任务，执行结束后从内存中消失。内核模块则是先在内核中注册自己以便服务于将来的某个请求，然后它的初始化函数结束，此时模块仍然存在于内核中，直到卸载函数被调用，模块才从内核中消失。

模块可选信息

1、许可证申明

宏MODULE_LICENSE用来告知内核，该模块带有一个许可证,没有这样的说明,加载模块时内核会抱怨。有效的许可证有"GPL“、"GPL v2"、"GPL and additional rights"、"Dual BSD/GPL"、"Dual MPL/GPL"和"Proprietary"。

模块可选信息

2、作者申明（可选）

```
MODULE_AUTHOR(“Simon Li");
```

3、模块描述（可选）

```
MODULE_DESCRIPTION("Hello World Module");
```

4、模块版本（可选）

```
MODULE_VERSION("V1.0");
```

5、模块别名（可选）

```
MODULE_ALIAS("a simple module");
```

模块可选信息

6、模块参数

通过宏module_param指定模块参数,模块参数用于在加载模块时传递参数给模块。

module_param(name,type,perm)

- name是模块参数的名称，type是这个参数的类型，
- perm是模块参数的访问权限。

type常见值:

bool:布尔型 int:整型 charp:字符串型

模块可选信息

perm 常见值:

S_IRUGO:任何用户都对/sys/module中出现的该参数具有读权限

S_IWUSR:允许root用户修改/sys/module中出现的该参数

例如:

```
int a = 3;
char *st;
module_param(a,int, S_IRUGO);
module_param(st,charp, S_IRUGO);
```

.ko PK .o

Before Linux 2.6, a user space program would interpret the ELF object(.o) file and do all the work of linking it to the running kernel, generating a finished binary image. The program would pass that image to the kernel and the kernel would do little more than stick it in memory. In Linux 2.6, the kernel does the linking. A user space program passes the contents of the ELF object file directly to the kernel. For this to work, the ELF object image must contain additional information. To identify this particular kind of ELF object file, we name the file with suffix ".ko"("kernel object") instead of ".o"

实例

模块申明

模块参数

实验

内核模块设计

- 使用模块参数
- 使用模块GPL申明、作者申明

内核符号导出

/proc/kallsyms 记录了内核中所有导出的符号的名字与地址。

什么叫导出？为什么要导出？

范例：符号导出

内核符号导出

内核符号的导出使用：

EXPORT_SYMBOL(符号名)

EXPORT_SYMBOL_GPL(符号名)

其中**EXPORT_SYMBOL_GPL**只能用于包含**GPL**许可证的模块。

实验

内核模块设计

- 设计两个内核模块，一个模块输出一些符号给另一模块

常见问题：版本不匹配

内核模块的版本由其所依赖的内核代码版本所决定，在加载内核模块时，**insmod**程序会将内核模块版本与当前正在运行的内核版本比较，如果不一致时，就会出现类似下面的错误：

```
insmod hello.ko
disagrees about version of symbol struct_module
insmod: error inserting 'hello.ko': -1 Invalid module format
```

常见问题：版本不匹配

解决方法：

1、使用 modprobe --force-modversion 强行插入

2、确保编译内核模块时，所依赖的内核代码版本等同于当前正在运行的内核。

可通过uname –r 察看当前运行的内核版本

内核打印

Printk是内核中出现最频繁的函数之一，通过将Printk与Printf对比，将有助于大家理解。

相同点：

- 打印信息

不同点：

- Printk在内核中使用，Printf在应用程序中使用
- Printk允许根据严重程度，通过附加不同的“优先级”来对消息分类。

内核打印

在<linux/kernel.h>中定义了8种记录级别。按照优先级递减的顺序分别是:

KERN_EMERG “<0>”

用于紧急消息,常常是那些崩溃前的消息。

KERN_ALERT “<1>”

需要立刻行动的消息。

KERN_CRIT “<2>”

严重情况。

KERN_ERR “<3>”

错误情况。

内核打印

- **KERN_WARNING “<4>”**
  有问题的警告

- **KERN_NOTICE “<5>”**
  正常情况,但是仍然值得注意

- **KERN_INFO “<6>”**
  信息型消息

- **KERN_DEBUG “<7>”**
  用作调试消息

内核打印

没有指定优先级的printk默认使用DEFAULT_MESSAGE_LOGLEVEL优先级，它是一个在kernel/printk.c中定义的整数。

在2.6.29内核中

```
#define DEFAULT_MESSAGE_LOGLEVEL 4
/* KERN_WARNING */
```

内核打印

控制台优先级配置

/proc/sys/kernel/printk

6 4 1 7

- **Console_loglevel**
- **Default_message_loglevel**
- **Minimum_console_level**
- **Default_console_loglevel**

Contents

内核启动

www.enjoylinux.cn

参考

《国嵌内核启动文档》