**主要内容：**

* 获取内核源码
* 内核源码的结构
* 编译内核的方法
* 内核开发的特点

### ****1. 获取内核源码****

内核是开源的，所有获取源码特别方便，参照以下的网址，可以通过git或者直接下载压缩好的源码包。

[http://www.kernel.org](http://www.kernel.org/)

你可以使用Git来获取最新提交到Linus版本树的一个副本:

$ git clone git://git.kernel.org/pub/scm/linux/kemel/git/torvalds/linux-2.6.git

### ****2. 内核源码的结构****

|  |  |
| --- | --- |
| **目录** | **说明** |
| arch | 特定体系结构的代码 |
| block | 块设备I/O层 |
| crypo | 加密API |
| Documentation | 内核源码文档 |
| drivers | 设备驱动程序 |
| firmware | 使用某些驱动程序而需要的设备固件 |
| fs | VFS和各种文件系统 |
| include | 内核头文件 |
| init | 内核引导和初始化 |
| ipc | 进程间通信代码 |
| kernel | 像调度程序这样的核心子系统 |
| lib | 同样内核函数 |
| mm | 内存管理子系统和VM |
| net | 网络子系统 |
| samples | 示例，示范代码 |
| scripts | 编译内核所用的脚本 |
| security | Linux 安全模块 |
| sound | 语音子系统 |
| usr | 早期用户空间代码（所谓的initramfs） |
| tools | 在Linux开发中有用的工具 |
| virt | 虚拟化基础结构 |

### ****3. 编译内核的方法****

**内核配置**

在编译内核之前，首先你必须配置它。这些配置项要么是二选一，要么是三选一。

内核提供了各种不同的工具来简化内核配置。

最简单的一种命令行工具：

$ make config

图形界面配置工具：

$ make menuconfig

基于默认的配置:

$ make def config

这些配置项会被存放在内核代码树根目录下的.config文件中。

**内核编译**

一旦内核配置好了（不论你是如何配置的），就可以使用一个简单的命令来编译它了 :

$ make

**内核安装**

在内核编译好之后，你还需要安装它。怎么安装就和体系结构以及启动引导工具(boot loader)息息相关了——这里不做介绍

**模块安装**

模块的安装是自动的，也是独立于体系结构的。以root身份，只要运行：

% make modules\_install

下面介绍3种删除那些不用的内核的方法：

#### **3.1 rpm 删除法**

   rpm -qa | grep kernel\*  (查找所有linux内核版本)  
   rpm -e kernel-(想要删除的版本)

#### **3.2 yum 删除法**

   yum remove kernel-(要删除的版本)

#### **3.3 手动删除**

   删除/lib/modules/目录下不需要的内核库文件  
   删除/usr/src/kernel/目录下不需要的内核源码  
   删除/boot目录下启动的核心档案禾内核映像  
   更改grub的配置，删除不需要的内核启动列表

### ****4. 内核开发的特点****

#### **4.1  无标准C库**

为了保证内核的小和高效，内核开发中不能使用C标准库

#### **4.2 使用GNU C，推荐用gcc 4.4或以后的版本来编译内核**

因为使用GNU C，所有内核中常使用GNU C中的一些扩展：

##### **4.2.1 内联函数**

内联函数示例：static **inline** void sample();

##### **4.2.2 内联汇编**

内联汇编用于偏近底层或对执行时间严格要求的地方。示例如下：

unsigned int low, high;

asm volatile("rdtsc" : "=a" (low), "=d" (high));/\* low 和 high 分别包含64位时间戳的低32位和高32位 \*/

##### **4.2.3 分支声明**

如果能事先判断一个if语句时经常为真还是经常为假，那么可以用unlikely和likely来优化这段判断的代码。

/\* 如果error在绝大多数情况下为0(假) \*/

if (unlikely(error)) {

/\* ... \*/

}

/\* 如果success在绝大多数情况下不为0(真) \*/

if (likely(success)) {

/\* ... \*/

}

#### **4.3 没有内存保护**

因为内核是最低层的程序，所以如果内核访问的非法内存，那么整个系统都会挂掉！

#### **4.4 不使用浮点数**

内核不能完美的支持浮点操作。

#### **4.5 内核栈容积小且固定**

内核栈的大小有编译内核时决定的，对于不用的体系结构，内核栈的大小虽然不一样，但都是固定的。

查看内核栈大小的方法：

ulimit -a | grep "stack size"

#### **4.6 同步和并发**

Linux是多用户的操作系统，所以必须处理好同步和并发操作，防止因竞争而出现死锁。

#### **4.7 可移植性**

Linux内核可用于不用的体现结构，支持多种硬件。所以开发时要时刻注意可移植性，尽量使用体系结构无关的代码。