**第二章 从内核出发**

**1. 获取Linux内核源码**

登录Linux内核官方网站 http://www.kernel.org，可以随时获取当前版本的内核源代码，可以是完整的压缩形式（使用tar命令创建一个压缩文件），也可以是增量补丁形式。内核压缩以GNU zip（gzip）和 bzip2两种形式发布。bzip2是默认和首选形式，因为压缩比gzip更有优势。以bzip2形式发布的Linux内核叫做linux-x.y.z.tar.bz2，这里x.y.z是内核源码的具体版本。

-c（create）：表示创建用来生成文件包

-x ：表示提取，从文件包中提取文件

-t ：可以查看压缩的文件

-z ：使用gzip方式进行处理，它与“C”结合就表示压缩，与“X”结合就表示解压缩

-j ：使用bzip2方式进行处理，它与“C”结合就表示压缩，与“X”结合就表示解压缩

-v（verbose） ：详细报告tar处理信息

-f（file）：表示文件，后面接着一个文件名

-C <指定目录> 解压指定目录

如果压缩形式是bzip2，则运行：

tar xvjf linux-x.y.z.tar.bz2

如果压缩形式是GNU的zip，则运行：

tar xvzf linux-x.y.z.tar.gz

**2.内核源码树**

|  |  |
| --- | --- |
| 目录 | 描述 |
| arch | 特定体系结构的源代码 |
| block | 块设备I/0层 |
| crypto | 加密API |
| Documentation | 内核源码文档 |
| drivers | 设备驱动层 |
| 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系统 |
| lib | 通用内核函数 |
| 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 |

**3.配置内核**

可配置选项以CONFIG\_FEATURE形式表示，其前缀为CONFIG。例如，对称多处理器（SMP）的配置选项为CONFIG\_SMP。如果设置了该选项，则SMP启用，否则，SMP不起作用。这些配置选项要么是二选一，要么是三选一。二选一就是yes或no。三选一可以是yes、no或module。module意味着改配置选项被选定了，但编译的时候这部分功能的实现代码是以模块（一种可以动态安装的独立代码段）的形式生产。在三选一的情况下，显然yes选项表示把代码编译进主内核映像中，而不是作为一个模块。驱动程序一般都是三选一的配置项。

配置选项也可以是字符串或整数。这些选项并不控制编译过程，而只是指定内核源码可以访问的值，一般以预处理宏的形式表示。

三种内核配置工具：

字符界面下的命令行工具：

make config（改工具会注意遍历所有的配置项，要求用户选择yes、no 或 module）

基于ncurse库编制的图形界面工具：

make menuconfig

基于gtk+的图形工具：

make gconfig

**4.内核开发的特点**

1.内核编程时既不能访问C库也不能访问标准的C头文件

2.内核编程时必须使用GNU C。

3.内核编程时缺乏像用户控件那样的内存保护机制。

4.内核编程时难以执行浮点运算。

5.内核给每个进程只有一个很小的定长堆栈。

6.由于内核支持异步中断、抢占和SMP，因此必须时刻注意同步和并发。

7.要考虑可移植性的重要性。

头文件：

基本的头文件位于内核源代码树顶级目录下的include目录中。例如，头文件<linux/inotify.h>对应内核源代码树的include/linux/inotify.h

内核代码无法调用printf函数，但是它提供的printk()函数几乎与printf()相同。printk()函数负责把格式化好的字符串拷贝到内核日志缓冲区上，这样，syslog程序就可以通过读取改缓冲来获取内核信息。syslogd会根据这个优先级标志类决定在什么地方显示这条系统信息。

**5.C语言扩展部分：**

1.内联（inline）函数：

用于消除函数调用和返回所带来的开销（寄存器存储和恢复）。会增加程序占用的空间或者占用更多的指令缓存。内核开发者通常把那些对时间要求比较高，而本身长度又比较短的函数定义成内联函数。

static inline void wolf(unsigned long tail\_size)

内联函数必须在使用时定义好，一般定义在头文件中，如果只在某个源文件中使用，也可以定义在源文件开始处。

2.内联汇编

gcc编译器支持在C函数中嵌入汇编指令。当然，在内核编程的时候，只有知道对应的体系结构，才能使用这个功能。

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 \*/

3.分支声明

对于条件选择语句，gcc内建了一条指令用于优化，在一个条件中经常出现，或者出现的时候很少，编译器可以根据这条指令对条件分支进行优化。内核把这条指令封装成了宏，比如likely()和unlikely()。

/\* 绝大多数时间 error 为 0\*/

if（unlikey(error)）

{

/\*\*\*\*\*\*\*\*\*\*\*/

}

/\* 绝大多数时间success 不为 0\*/

if（unlikey(success)）

{

/\*\*\*\*\*\*\*\*\*\*\*/

}