Chinese translated version of Documentation/sparse.txt

If you have any comment or update to the content, please contact the original document maintainer directly. However, if you have a problem communicating in English you can also ask the Chinese maintainer for help. Contact the Chinese maintainer if this translation is outdated or if there is a problem with the translation.

Chinese maintainer: Li Yang <leo@zh-kernel.org>

Documentation/sparse.txt 的中文翻译

如果想评论或更新本文的内容，请直接联系原文档的维护者。如果你使用英文交流有困难的话，也可以向中文版维护者求助。如果本翻译更新不及时或者翻译存在问题，请联系中文版维护者。

中文版维护者： 李阳 Li Yang <leo@zh-kernel.org>
中文版翻译者： 李阳 Li Yang <leo@zh-kernel.org>

以下为正文

使用 sparse 工具做类型检查

"__bitwise" 是一种类型属性，所以你应该这样使用它：

```
typedef int __bitwise pm_request_t;

enum pm_request {
        PM_SUSPEND = (__force pm_request_t) 1,
        PM_RESUME = (__force pm_request_t) 2
};
```

这样会使 PM_SUSPEND 和 PM_RESUME 成为位方式(bitwise)整数（使用"__force"是因为 sparse 会抱怨改变位方式的类型转换，但是这里我们确实需要强制进行转换）。而且因为所有枚举值都使用了相同的类型，这里的"enum pm_request"也将会使用那个类型做为底层实现。

而且使用 gcc 编译的时候，所有的 __bitwise/__force 都会消失，最后在 gcc 看来它们只不过是普通的整数。

坦白来说，你并不需要使用枚举类型。上面那些实际都可以浓缩成一个特殊的"int __bitwise"类型。

所以更简单的办法只要这样做：

```
typedef int __bitwise pm_request_t;

#define PM_SUSPEND ((__force pm_request_t) 1)
#define PM_RESUME ((__force pm_request_t) 2)
```

现在你就有了严格的类型检查所需要的所有基础架构。

一个小提醒：常数整数"0"是特殊的。你可以直接把常数零当作位方式整数使用而不用担心 sparse 会抱怨。这是因为"bitwise"（恰如其名）是用来确保不同位方式类型不会被弄混（小尾模式，大尾模式，cpu尾模式，或者其他），对他们来说常数"0"确实是特殊的。

获取 sparse 工具

你可以从 Sparse 的主页获取最新的发布版本：

        http://www.kernel.org/pub/linux/kernel/people/josh/sparse/

或者，你也可以使用 git 克隆最新的 sparse 开发版本：

        git://git.kernel.org/pub/scm/linux/kernel/git/josh/sparse.git

DaveJ 把每小时自动生成的 git 源码树 tar 包放在以下地址：

        http://www.codemonkey.org.uk/projects/git-snapshots/sparse/

一旦你下载了源码，只要以普通用户身份运行：

        make
        make install

它将会被自动安装到你的 ~/bin 目录下。

使用 sparse 工具

用"make C=1"命令来编译内核，会对所有重新编译的 C 文件使用 sparse 工具。或者使用"make C=2"命令，无论文件是否被重新编译都会对其使用 sparse 工具。如果你已经编译了内核，用后一种方式可以很快地检查整个源码树。

make 的可选变量 CHECKFLAGS 可以用来向 sparse 工具传递参数。编译系统会自动向 sparse 工具传递 -Wbitwise 参数。你可以定义 __CHECK_ENDIAN__ 来进行大小尾检查。

        make C=2 CHECKFLAGS="-D__CHECK_ENDIAN__"

这些检查默认都是被关闭的，因为他们通常会产生大量的警告。