内核使用的数据类型主要被分为三类：

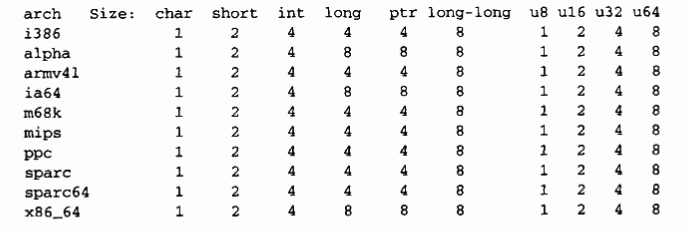
类似int的c语言类型

类似u32的确定大小的类型

类型pid\_t的内核对象类型

1. **c言语类型**

下图是数据类型在各个平台上的大小



我们不能假设c言语类型的大小，例如long在某些平台上是4byte，某些平台是8byte

内存地址（指针）的类型是unsigned long类型

1. **确定大小的数据类型**

如果我们想使用确定大小的类型，则应使用如下类型

头文件 <asm/types.h>

u8; /\* unsigned byte (8 bits) \*/

u16; /\* unsigned word (16 bits) \*/

u32; /\* unsigned 32-bit value \*/

u64; /\* unsigned 64-bit value \*/

1. **内核类型**

内核类型通常以 \_t 结尾，很多 \_t 类型在 <linux/types.h> 中定义

现在许多开发者不在喜欢用typedef定义新类型了

1. **其他移植性问题**

**时间间隔**

HZ表示1秒

**页大小**

一页大小未PAGE\_SIZE字节, 而不是 4KB

**字节序**

同样，不要假设cpu使用的字节序

u32 cpu\_to\_le32 (u32);  // 将32位cpu使用的值转为小头字节序

u32 le32\_to\_cpu (u32);  // 将32位小头字节序转为cpu使用的值

**数据对齐**

指针是以4的倍数进行对齐，如果指针不是4的整数倍，要访问指针的值会出现错误，可使用如下的宏

#include <asm/unaligned.h>

get\_unaligned(ptr);

put\_unaligned(val, ptr);

结构数据对齐

struct

{

 u16 id;

 u64 lun;

 u16 reserved1;

 u32 reserved2;

}

\_\_attribute\_\_ ((packed)) scsi;

如果没有 \_\_attribute\_\_ ((packed)), lun 成员可能被在前面添加 2 个填充者字节或者 6 个（如果我们在 64-位平台上编译这个结构）

1. **链表**

链表是内核经常使用的数据类型，详见《Linux内核设计与实现》->《6 内核数据结构》