文档说明	作者	日期
来自微信公众号：宅学部落(armLinuxfun)	wit	2020.3.6
嵌入式视频淘宝店：https://wanglitao.taobao.com/		
作者微信：brotau（宅学部落）		

枚举（enum）是C语言的一种特殊类型。当我们在编程中遇到定义一些固定长度或范围的数值时，可以考虑使用枚举类型。使用枚举可以让我们的程序可读性更强、看起来更加直观。举个例子，如果我们在编程中需要使用数字0~6分别表示星期日~星期六，程序的可读性就不高，我们需要翻手册或者看程序注释才能知道每个数字具体代表什么意思。如果我们使用枚举呢，基本上不需要看注释或手册就可知晓其大意。

```
enum week    // enum 枚举类型{枚举值列表};
{
    SUN,MON,TUE,WED,THU,FRI,SAT,
};
enum week today = SUN; //使用枚举类型定义一个变量
```

使用enum定义的枚举值列表中，默认值是从0开始，然后依次递增：SUN=0，MON=1...。当然我们也可以显式指定枚举值：

```
enum week
{
    SUN = 1,MON,TUE,WED,THU = 7,FRI,SAT,
};
//SUN=1，那么接下来MON=2，TUE=3，WED=4
//THU=7，那么接下来FRI=8，SAT=9
```

1. enum经常使用的三种方法

使用枚举类型定义变量，使用方法跟结构体、共用体类似，经常使用的三种方法如下：

```
enum week //定义枚举类型的同时,定义枚举变量
{
    SUN,MON,TUE,WED,THU,FRI,SAT,
}today, tomorrow;

enum //可以省去枚举类型名，直接定义变量
{
    SUN,MON,TUE,WED,THU,FRI,SAT,
}today, tomorrow;
```

```
enum week //先定义枚举类型，再定义枚举变量
{
    SUN,MON,TUE,WED,THU,FRI,SAT,
};
enum week today, tomorrow;
```

2. 枚举的本质

在C语言中，枚举是一种类型，属于整型的范畴，使用enum定义的枚举值列表，其实就是从0开始的一系列整数序列。整型除了short、int、long、long long外，还包括char、_Bool（C99标准新增）和enum。因此，枚举的使用其实和整数值其实没啥区别：我们使用枚举类型定义的变量，同样可以作为函数参数、函数返回值、用来定义数组、甚至和结构体混用等。

```
enum week get_week_time (void);
int set_week_time (enum week time_set);
int change_week_time (enum week *p);
enum week a[10];
struct student
{
    char name[20];
    int age;
    enum week birthday;
};
```

枚举有点类似于typedef，给一个数值添加一个别名，让我们的程序更加直观、可读性更高。枚举类型在本质上就是有命名的整数，属于整型的一种，在代码中是可以和整型互换的。

```
enum week t = SUN;
int  t2 = SUN;
enum week t3 = t2;
enum week t4 = 100;
```

在上面的代码中，枚举变量和整型变量相互赋值，都是可以正常编译和运行的。我们在代码中使用枚举类型，在最终编译生成的可执行文件中都会被整型数值代替。

```
enum week
{
    SUN = 5,MON,TUE,WED,THU,FRI,SAT,
};

int main (void)
{
    enum week today = THU;
    return 0;
}
```

在上面的示例代码中，我们定义了一个枚举类型week，然后定义了一个枚举变量today，并赋值为THU。反汇编上面的代码，我们可以看到汇编代码：

```
00010400 <main>:
   10400:   e52db004    push    {fp}
   10404:   e28db000    add fp, sp, #0
   10408:   e24dd00c    sub sp, sp, #12
   1040c:   e3a03009    mov r3, #9
   10410:   e50b3008    str r3, [fp, #-8]
   10414:   e3a03000    mov r3, #0
   10418:   e1a00003    mov r0, r3
   1041c:   e24bd000    sub sp, fp, #0
   10420:   e49db004    pop {fp}        ;
   10424:   e12fff1e    bx  lr
```

在C程序中定义的枚举变量today，在汇编代码的第1040c处，我们可以看到：枚举值THU被替换为整型数值9。使用枚举的唯一好处就是增加代码的可读性，它的作用跟宏定义的作用有异曲同工之妙。

3. 枚举和宏

枚举与预处理指令#define的作用差不多，都是为了增加代码的可读性。但在实际使用中，两者还是有些差别的：宏在预处理阶段，通过简单的字符串替换就全部被替换掉了，编译器根本不知道有宏这么一个玩意；而枚举类型则在编译阶段全部替换为整型。

跟宏相比，枚举的优势是：枚举可以自动赋值，而宏则需要一个一个单独定义。因此，在自定义一些有规则的类型值的时候，使用枚举会更加方便。枚举可以自定义的变量值来代替数字值，使我们的程序代码有更高的可读性。

4. Linux内核中的枚举类型

在Linux内核代码中，充斥着大量的枚举类型数据，有些枚举类型的定义看起来很奇怪，比如：

```
enum
{
    MM_FILEPAGES,
    MM_ANONPAGES,
    MM_SWAPENTS,
    NR_MM_COUNTERS
};
enum pid_type
{
    PIDTYPE_PID,
    PIDTYPE_PGID,
    PIDTYPE_SID,
    PIDTYPE_MAX
};
```

Linux内核中使用enum定义的枚举类型大部分是没有枚举名的，而且通常会在一串枚举值之后带上一个NR*的元素用来表示枚举值的数量。当我们不需要使用枚举类型去定义一个枚举变量时，枚举并不需要一个名字，这些无名的枚举类型其实就相当于宏定义。而最后一个元素NR或MAX，一般可以用来记载枚举列表中元素的个数，或者作为循环判断的边界值。

5. 使用枚举需要注意的地方

什么是类型？类型是一定范围的数值及方法的集合。枚举作为整型类型的一种，在编程使用过程中，也有一些注意的地方，比如作用域。使用枚举定义的常量也遵循数据作用域规：包括文件作用域、代码块作用域等，在同一个作用域能不能出现重名的枚举常量名。

```
enum week1
{
    SUN,MON,TUE,WED,THU,FRI,SAT,
};
enum week2
{
    SAT,UNKNOW,
};
int main (void)
{
    return 0;
}
```

在上面的代码中，我们定义了两个枚举类型，其中枚举常量SAT重名，编译时就会发生如下错误：

```
error: redeclaration of enumerator `SAT'
error: previous definition of 'SAT' was here
```

出现错误的原因是，我们定义的不同枚举类型中的两个枚举常量名同在一个作用域：文件作用域，我们稍微改一下代码就可以避免冲突：

```
#include <stdio.h>
enum week1
{
    SUN,MON,TUE,WED,THU,FRI,SAT,
};
int main (void)
{
    printf("%d\n", SAT);
    enum week2
    {
        SAT,UNKNOW,
    };
    printf("%d\n", SAT);
    return 0;
}
```

我们将枚举类型week2的定义放到了main函数内，week2的作用域就从文件作用域变为代码块作用域。这个时候，两个枚举类型中的同名枚举常量就不会再发生冲突，程序的运行结果为：

```
6
0
```


ARM