

神奇的__attribute__



秦明Qinmin (/u/fff74d0ebed7) [+ 关注](#)

2016.04.12 11:52* 字数 1605 阅读 2668 评论 1 喜欢 23

(/u/fff74d0ebed7)

`__attribute__`是GNU C特色之一,在iOS用的比较广泛。如果你没有用过,那系统库你总用过,在Foundation.framework中有很多地方用到`__attribute__`特性。`__attribute__`可以设置函数属性 (Function Attribute)、变量属性 (Variable Attribute) 和类型属性 (Type Attribute)。接下来就从iOS中常见用法谈起。

1. format

作用：编译器会检查格式化字符串与“...”的匹配情况，防止产生难以发现的Bug。

用法：

```
__attribute__((format(printf,m,n)))
```

```
__attribute__((format(scanf,m,n)))
```

其中参数m与n的含义为：

m 格式化字符串 (format string) 的位置 (顺序从1开始)；

n 参数“...”的位置 (顺序从1开始)；

例子：



```
FOUNDATION_EXPORT void NSLog(NSString *format, ...) NS_FORMAT_FUNCTION(1,2);
#define NS_FORMAT_FUNCTION(F,A) __attribute__((format(__NSString__, F, A)))
```

```
#define kMaxStringLen 512

extern void MyLog(const char *tag,const char *format,...) __attribute__((format(printf, 2, 3)));

void MyLog(const char *tag,const char *format,...) {
    va_list ap;
    va_start(ap, format);

    char* pBuf = (char*)malloc(kMaxStringLen);
    if (pBuf != NULL)
    {
        vsnprintf(pBuf, kMaxStringLen, format, ap);
    }
    va_end(ap);

    printf("TAG:%s Message:%s", tag,pBuf);

    free(pBuf);
}
```

2. deprecated

作用：使编译会给出过时的警告。

用法：

`__attribute__((deprecated))`

`__attribute__((deprecated(s)))`

例子：



```
#define DEPRECATED_ATTRIBUTE __attribute__((deprecated))
#if __has_feature(attribute_deprecated_with_message)
#define DEPRECATED_MSG_ATTRIBUTE(s) __attribute__((deprecated(s)))
#else
#define DEPRECATED_MSG_ATTRIBUTE(s) __attribute__((deprecated))
#endif
```

3. availability

作用：指明API版本的变更。

用法：

`__attribute__((availability(macosx,introduced=m,deprecated=n)))`

m 引入的版本

n 过时的版本

例子：

```
#define CF_DEPRECATED_IOS(_iosIntro, _iosDep, ...) __attribute__((availability(ios,i
```

4. unavailable

作用：告诉编译器该方法不可用，如果强行调用编译器会提示错误。比如某个类在构造的时候不想直接通过init来初始化，只能通过特定的初始化方法，就可以将init方法标记为unavailable。

用法：

`__attribute__((unavailable))`

例子：



```
#define UNAVAILABLE_ATTRIBUTE __attribute__((unavailable))

#define NS_UNAVAILABLE UNAVAILABLE_ATTRIBUTE
```

```
#import <Foundation/Foundation.h>

@interface Person : NSObject

@property(nonatomic,copy) NSString *name;


@property(nonatomic,assign) NSInteger age;


- (instancetype)init NS_UNAVAILABLE;

- (instancetype)initWithName:(NSString *)name age:(NSInteger)age;

@end
```

```
Person *person = [[Person alloc] init];
```

 Unused variable 'person'

 'init' is unavailable

066226DB-F65E-4763-B21A-7D8E04B151FF.png

5. const

作用：用于带有数值类型参数的函数上。当重复调用带有数值参数的函数时，由于返回值是相同的，所以此时编译器可以进行优化处理，除第一次需要运算外，其它只需要返回第一次的结果就可以了，进而可以提高效率。该属性主要适用于没有静态状态和副作用的一些函数，并且返回值仅仅依赖输入的参数。（const参数不能用在带有指针类型参数的函数中，因为该属性不但影响函数的参数值，同样也影响到了参数指向的数据，它可能会对代码本身产生严重甚至是不可恢复的严重后果。）

用法：

```
__attribute__((const))
```

例子：



```
int __attribute__((const)) add(int x)
{
    printf("%s(%d)\n", __FUNCTION__, x);
    return x + 1;
}

int add2(int x)
{
    printf("%s(%d)\n", __FUNCTION__, x);
    return x + 1;
}

int main(int argc, char* argv[])
{
    int i, j;

    i = add(10);
    j = add(10);

    printf("%d %d\n", i, j);

    i = add2(10);
    j = add2(10);

    printf("%d %d\n", i, j);

    return 0;
}
```

6. cleanup

作用：离开作用域之后执行指定的方法。实际应用中可以在作用域结束之后做一些特定的工作，比如清理。

用法：__attribute__((cleanup(...)))

例子：



```
static void stringCleanUp(__strong NSString **string) {
    NSLog(@"%@", *string);
}

void testCleanUp() {
    __strong NSString *string __attribute__((cleanup(stringCleanUp))) = @"stringClea
}

static void blockCleanUp(__strong void(^ *block)()) {
    if (*block) {
        (*block)();
    }
}

void testBlockCleanUp() {
    __strong void(^block)() __attribute__((cleanup(blockCleanUp))) = ^{
        NSLog(@"block");
    };
}

static void lambdaCleanUp(void (**lambda)()) {
    if (*lambda) {
        (*lambda)();
    }
}

void testLambdaCleanUp() {
    void (*lambda)() __attribute__((cleanup(lambdaCleanUp))) = []() {
        puts("lambda");
    };
}

int main(int argc, char * argv[]) {
    @autoreleasepool {
        testCleanUp();

        testBlockCleanUp();

        testLambdaCleanUp();
    }
    return 0;
}
```



```
//结合宏定义使用
#define BlockCleanUp __strong void(^block)() __attribute__((cleanup(blockCleanUp)))
#define LambdaCleanUp void (*lambda)() __attribute__((cleanup(lambdaCleanUp))) = []()
void testDefine() {
    BlockCleanUp {
        puts("BlockCleanUp");
    };

    LambdaCleanUp{
        puts("LambdaCleanUp");
    };
}
```

7. constructor与destructor

作用：__attribute__((constructor)) 在main函数之前执行,__attribute__((destructor)) 在main函数之后执行。__attribute__((constructor(PRIORITY)))和__attribute__((destructor(PRIORITY)))按优先级执行。（可用于动态库注入的Hook）

用法：

__attribute__((constructor))

__attribute__((destructor))

__attribute__((constructor(PRIORITY)))

__attribute__((destructor(PRIORITY)))

PRIORITY 为优先级

例子：



```
void __attribute__((constructor)) start() {  
    NSLog(@"%s", __FUNCTION__);  
}  
  
void __attribute__((destructor)) end() {  
    NSLog(@"%s", __FUNCTION__);  
}  
  
![[Uploading E71D5B89-60DF-47C6-A923-F731680F25B6_088963.png . . .]]int main(int argc,  
  
    NSLog(@"%s", __FUNCTION__);  
  
    return 0;  
}
```

```
Test[1026:228110] start  
Test[1026:228110] main  
Test[1026:228110] end
```

E71D5B89-60DF-47C6-A923-F731680F25B6.png




```
void __attribute__((constructor)) start() {
    NSLog(@"%s",__FUNCTION__);
}

void __attribute__((constructor(100))) start100() {
    NSLog(@"%s",__FUNCTION__);
}

void __attribute__((constructor(101))) start101() {
    NSLog(@"%s",__FUNCTION__);
}

void __attribute__((destructor)) end() {
    NSLog(@"%s",__FUNCTION__);
}

void __attribute__((destructor)) end100() {
    NSLog(@"%s",__FUNCTION__);
}

void __attribute__((destructor)) end101() {
    NSLog(@"%s",__FUNCTION__);
}

int main(int argc, char * argv[]) {

    NSLog(@"%s",__FUNCTION__);

    return 0;
}
```

```
2016-04-11 21:09:58.266 Test[1034:229125] start100
2016-04-11 21:09:58.268 Test[1034:229125] start101
2016-04-11 21:09:58.268 Test[1034:229125] start
2016-04-11 21:09:58.268 Test[1034:229125] main
2016-04-11 21:09:58.268 Test[1034:229125] end101
2016-04-11 21:09:58.268 Test[1034:229125] end100
2016-04-11 21:09:58.268 Test[1034:229125] end
```

46182749-3804-40A2-ACA6-691BB2E22B71.png

8. noreturn

作用：定义有返回值的函数时，而实际情况有可能没有返回值，此时编译器会报错。加上**attribute((noreturn))**则可以很好的处理类似这种问题。



用法：

`__attribute__((noreturn))`

例子：

```
void __attribute__((noreturn)) onExit();

int test(int state) {
    if (state == 1) {
        onExit();
    } else {
        return 0;
    }
}
```

9. nonnull

作用：编译器对函数参数进行NULL的检查

用法：`__attribute__((nonnull(...)))`

```
extern void *my_memcpy_2 (void *dest, const void *src, size_t len) __attribute__((no
extern void *my_memcpy_3 (void *dest, const void *src, const void *other, size_t len

void test_my_memcpy() {
    my_memcpy_2(NULL, NULL, 0);
    my_memcpy_3("", "", NULL, 0);
}
```



```

22 extern void *my_memcpy_2 (void *dest, const void *src, size_t len) __attribute__((nonnull
    (1, 2)));
23
24 extern void *my_memcpy_3 (void *dest, const void *src, const void *other, size_t len)
    __attribute__((nonnull (1, 2, 3)));
25
26 void test_my_memcpy() {
27     my_memcpy_2(NULL, NULL, 0);
28     my_memcpy_3("", "", NULL, 0);
29 }
30

```

Null passed to a callee that requires a non-null argument

Null passed to a callee that requires a non-null argument

1CA959CA-3710-4F9B-AFC5-A4F263811F6D.png

10. aligned 与 packed

作用：*aligned(m)* 将强制编译器尽其所能地确保变量在分配空间时采用m字节对齐方式。*packed*该属性对struct 或者union 类型进行定义，设定其类型的每一个变量的内存约束，当用在enum 类型定义时，暗示了应该使用最小完整的类型。*aligned* 属性使被设置的对象占用更多的空间，使用*packed* 可以减小对象占用的空间。

用法：*attribute ((aligned (m)))*

attribute ((aligned))

attribute ((packed))

例子：



```
//运行在iPhone5模拟器上
struct p {
    int a;
    char b;
    short c;
}__attribute__((aligned(4))) pp;

struct m {
    char a;
    int b;
    short c;
}__attribute__((aligned(4))) mm;

struct o {
    int a;
    char b;
    short c;
}oo;

struct x {
    int a;
    char b;
    struct p px;
    short c;
}__attribute__((aligned(8))) xx;

struct MyStruct {
    char c;
    int i;
    short s;
}__attribute__((__packed__));

struct MyStruct1 {
    char c;
    int i;
    short s;
}__attribute__((aligned));

struct MyStruct2 {
    char c;
    int i;
    short s;
}__attribute__((aligned(4)));

struct MyStruct3 {
    char c;
    int i;
    short s;
}__attribute__((aligned(8)));
```



```
struct MyStruct4 {
    char c;
    int i;
    short s;
}__attribute__((aligned(16)));

int main(int argc, char * argv[]) {

    printf("sizeof(int)=%lu,sizeof(short)=%lu,sizeof(char)=%lu\n", sizeof(int), sizeof(short), sizeof(char));

    printf("pp=%lu,mm=%lu \n", sizeof(pp), sizeof(mm));

    printf("oo=%lu,xx=%lu \n", sizeof(oo), sizeof(xx));

    printf("mystruct=%lu \n", sizeof(struct MyStruct));

    printf("mystruct1=%lu \n", sizeof(struct MyStruct1));

    printf("mystruct2=%lu \n", sizeof(struct MyStruct2));

    printf("mystruct3=%lu \n", sizeof(struct MyStruct3));

    printf("mystruct4=%lu \n", sizeof(struct MyStruct4));

    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

```
sizeof(int)=4,sizeof(short)=2,sizeof(char)=1
pp=8,mm=12
oo=8,xx=24
mystruct=7
mystruct1=16
mystruct2=12
mystruct3=16
mystruct4=16
```

DD0CE993-E373-410E-B0CC-5F91C0E56729.png

参考资料：

http://nshipster.com/__attribute__/ (http://nshipster.com/__attribute__/)





秦明Qinmin (/u/fff74d0ebed7)

写了 60131 字，被 582 人关注，获得了 501 个喜欢

(/u/fff74d0ebed7)

+ 关注

iOS工程师一枚，喜欢尝试新的事物。

♡ 喜欢 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-like-button) | 23



更多分享

(http://cwb.assets.jianshu.io/notes/images/3537936/weibo/image_C

被以下专题收入，发现更多相似内容



程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)



iOS,objc... (/c/3c1130eb8317?utm_source=desktop&utm_medium=notes-included-collection)



首页投稿 (/c/bDHhpK?utm_source=desktop&utm_medium=notes-included-collection)



君赏博客 (/c/ffeb0e8caae4?utm_source=desktop&utm_medium=notes-included-collection)



技术之道 (/c/687dfb38e8e6?utm_source=desktop&utm_medium=notes-included-collection)



预编译处理 (/c/8af95297f918?utm_source=desktop&utm_medium=notes-included-collection)



iOS开发成长之路 (/c/9735671a2cbb?utm_source=desktop&utm_medium=notes-included-collection)



