一、常见的错误信息

1. error: 'xxx' undeclared(first use in this function)

还没有定义（在此函数中第一次使用）。

注：使用某一个变量时，如果使用前还没有定义，会出现该错误。在oc以及c语言中，使用变量前必须先定义它。

这个错误经常出现在忘记进行变量定义的情况下。但是，慢慢拿习惯后，这种错误会出现得很少。反而经常出此种错误的原因是变量名拼写错误，即出现使用的变量名与定义的变量名不一致的情况。

例：

```
- (void)test
{
    int count = 0;
    return conut + 1;//出现变量count与conut不一致。 }
```

正确的写法：

```
- (void)test
{
    int count = 0;
    return count + 1;//将变量名count修改成一致。}
```

2. error: parse error before 'xxx' token

在‘xxx’之前发生了解析错误。

注：这个是在发生低级失误时才会出现的错误。解析错误的意思是程序语句以oc的语法不能解析。仔细看看发生错误的地方，一定会发现不符合语法的部分。

例：

```
- (void)test
{
    NSString *str = @"This is test" //此处遗漏了分号
     NSLog(str);
}
```

3. error: invalid preprocessing directive #xxx

关键字#xxx不正确。

注：当#include，#import等以#开始的关键字出现拼写错误时，会显示这种错误信息。在Xcode中，紧随#后输入的字符串都会变色，非常容易发生错误。

例：

```
#improt <Cocoa/Cocoa.h>  //#import拼写错误
```

4. error: xxx.h: No Such file or directory

名为xxx.h的文件或目录不存在。

注：在#include，#import中指定的文件不存在时会显示此种信息。最可能的原因是，文件名输入错误，好好检查一下文件名。

如果发生这种错误，最好也确认一下实际文件。因为还有可能是文件的检索路径没有指定正确。只要文件是包含在工程的文件夹中都是没有问题的。

例：

```
#import "AppContrller.h" //文件AppContrller.h是不存在的。
```

正确的是：

```
#import AppContrller.h
```

5. error: Undefined symbols: 'xxx'

符号‘xxx’没有被定义。

注：此错误不仅在编译时发生，在连接时也会发生。连接时使用了原本不存在的类或者函数时会出现此信息。

最经常出现的是，函数名出现输入错误。调用c语言函数的时候，就算名称错误，编译也会通过的。但是在连接的时候就会出现此错误的信息。

其他可能的原因是，使用Cocoa以外的框架或者库时，这些框架或者库没有包含进工程中。需要的库或者框架都必须包含到工程中。

二、常见的警告信息

警告中有绝对需要修正的，也有完全不用理会的。但是，将所有的警告都消去还是让人比较舒服的，所以还是要尽量修改。

1. warning: 'xxx' may not respond to 'yyy'

类‘xxx’中没有方法‘yyy’的声明。

注：当调用某一类中的方法时，类声明中并没有包含此方法出现时此信息。首先可能的原因是，方法名输入错误，请仔细检查一下方法名称，确保正确。

例：

```
NSString *str;
str = [NSString stringWithForatm:@"%d", 10];//方法名称错误。
```

正确的是：

```
NSString *str;
str = [NSString stringWithFormat:@"%d", 10];
```

另外，在某一类调用自己定义的方法时，如果方法都追加在类声明中的话，不会出现任何问题。如果实际调用的地方在方法定义的前方，也会出现这种警告信息。这是因为编译器对方法定义的检查是从文件的开始处顺序经行的。利用这个特性，如果不想其他类调用方法，可以不用追加在类声明中。

例：
有警告：

```
- (void)methodA
{
    [self methodB];//methodB的定义在后面
}

- (void)methodB
{
}
```

无警告：

```
- (void)methodB
{
}

- (void)methodA
{
    [self methodB];//methodB的定义在前时，不出现警告
}
```

如果不理会这个警告会出现什么情况？首先编译与连接是能通过的，因此应用程序时能够启动的。但应用程序实际运行到此处时，才会检查调用的方法到底是否真的在类中定义。如果没有定义则抛出异常，否则正常执行通过。因此，如果确实已经在类中定义了这个方法，可以故意忽略此警告。

2. warning: unused variable 'xxx'

变量‘xxx’没有被使用。

注：变量已经定义了，但是一次都没有被使用时出现此信息。经常出现的是，曾经使用的变量，经过修改后不再使用它了，但定义还保存着。此时，只用删除变量的定义即可。不删也可以。

另外，定义的变量名与使用的变量名不一致时，也会出现这个警告信息。

例：

```
- (void)test
{
    int a, b;//b没有被使用
    a = 5;
    return a;
}
```

正确的是：

```
- (void)test
{
    int a;//将b的定义删除
    a = 5;
    return a;
}
```

3. warning: local declaration of 'xxx' hides instance variable

本地变量‘xxx’覆盖了实例变量（即同名）。

注：当方法中定义的变量名与实例变量的某个变量同名，就会显示这个警告信息。因为同名，所有有一方将不能访问。 这时外部的实例变量将不能被访问，将一方变量名修改后即可。

例：

```
@interface MyObject:NSObject
{
    int count;
}
@end

@implementation MyObject

- (void)updateCount:(int)count
{
    //参数的名称与实例变量名相同。
}
```

正确的是：

```
@interface MyObject:NSObject
{
    int count;
}
@end

@implementation MyObject

- (void)updateCount:(int)num
{
    //修改参数的名称，使它与实例变量名不同。
}
```

4. warning: incomplete implementation of class 'xxx'
   warning: method definition for 'yyy' not found
   类‘xxx’的代码编写没有完成。
   方法‘yyy’的定义没有找到。

注：没有给类声明中的某个方法编写执行代码时，会显示此警告信息。出现警告后，该完成的执行代码应该完成，如果觉得这个方法不需要了，可以在类声明文件中删除此方法的定义。另外，如果实际代码处的方法名与定义的方法名出现不一致时，也会出现此警告信息。

5. warning: control reaches end of non-void function
   非void类型的函数没有设置返回值。

注：方法或函数需要返回值的情况下，没有设置任何返回值时出现的警告信息。 返回值类型为void以外的方法中，请务必返回一个具体的值。如果不需要返回值，请将方法的返回值类型修改为void。

相反，如果返回值设置为void类型，而在函数或方法中返回了某个值的时候，会显示 "'return' with a value, in function returning void（返回void的函数中，返回了值）"的警告。

例：

```
- (int)test:(int)count
{
    count++;//需要返回整型值而没有返回任何值
}
```

正确的是：

```
- (int)test:(int)count
{
```

```
    return count++;
}
```

6. warning: passing argument n of 'xxx' assignment from distinct Objective-C type

方法‘xxx’的第n个参数与Objective-C的类型不一致。

注：向方法‘xxx’中传递参数时，传递过来的参数对象与方法中声明的参数类型不一致时，会出现此警告。例如，声明的是NSEnumerator类型，传递进来的为NSString类型，则显示此警告。

最可能发生的原因是，方法的参数较多，设置时将顺序弄错了。在使用参数较多的方法时，出现这个警告信息的情况下，请仔细检查一下参数的顺序。

另外，在定义方法时可以利用这个警告。如果想将传递过来的参数指定为特定的类时，给参数定义明确的类型。如果任何对象都可以的情况，则定义为id类型。参数的定义包含了类设计者给使用者的信息。

例：

```
int value = 3;
NSString *str;
str = [NSString stringWithFormat:"%d", value];
//stringWithFormat:的参数不能是c语言的字符串。
```

正确的是：

```
int value = 3;
NSString *str;
str = [NSString stringWithFormat:"@%d", value];
//在参数字符串前追加@符号
```