-W 和 -Wall 的作用：打印编译时的警告信息

gcc 编译默认是 -w ：关闭所有警告信息

-Wall ： 所有的意思

-Wall开启“所有”的警告。强烈建议加上，并推荐该选项成为共识。如case语句没有default处理，有符号、无符号处理，未使用变量(特别是函数有大量未使用的数组，占用栈空间，测试发现，开辟一个未使用的8MB的数组，程序有coredump)，用%d来打印地址，或%s打印int值，等，都可以发出警告。

-Wchar-subscripts：

使用char类作为数组下标(因为char可能是有符号数)

-Wcomment：

注释使用不规范。如“/* */”注释中还包括“/*”。我在项目源码发现过，不止一处。

-Wmissing-braces

括号不匹配。在多维数组的初始化或赋值中经常出现。下面a没有完整被初始化，b完整初始化：

```
int a[2][2] = { 0, 1, 2, 3 };
int b[2][2] = { { 0, 1 }, { 2, 3 } };
```

-Wparentheses

括号不匹配，在运算符操作或if分支语句中，可能会出现此警告。

如“a&&b||c^d”会出现警告。下面代码片段也会有警告

```
{
if (a)
  if (b)
    foo ();
else
  bar (); // 这个else实际是if (b)的分支，不是if (a)，因此，要用括号来表明其属于哪个分支
}
```

这类bug隐藏得深，建议显式地加上括号。

-Wsequence-point

如出现i=i++这类代码，则报警告。-Wall默认有该警告

-Wswitch-defaultcase
没有default时，报警告

-Wunused-but-set-parameter
设置了但未使用的参数警告

-Wunused-but-set-variable
设置了但未使用的变量警告

-Wunused-function
声明但未使用函数

-Wunused-label
未使用的标签，比如用goto会使用label，但在删除goto语句时，忘了删除label。

-Wunused-variable
未使用的变量

-Wmaybe-uninitialized
变量可能没有被初始化。特别是在有if语句或switch语句中，最好在声明变量时加上初始化。
下面代码片段中，当y不是1、2、3时，x没有明确的值，是不安全的。

```
{
int x;
switch (y)
  {
  case 1: x = 1;
    break;
  case 2: x = 4;
    break;
  case 3: x = 5;
  }
foo (x);
```

```
}
```

-Wfloat-equal
对浮点数使用等号，这是不安全的。

```
{
float d = 2.0;
if (d == i)
{
  ...
}
}
```

-Wreturn-type
函数有返回值，但函数体个别地方没有返回值(特别是有if判断，可能忘记在else添加返回值)。

```
int foo()
{
 if(a==1)
 {
      return ok;
 }
 // no return here
}
```

-Wpointer-sign
指针有符号和无符号的错误传参。如函数使用unsigned char*，但传入char*指针。

-Wsign-compare
有符号和无符号比较。

-Wconversion-null

-Wsizeof-pointer-memaccess
在sizeof中经常出现，下面代码片段中，this为指针，4字节，无法保证完整初始化类。

```
memset(this, 0, sizeof(this));
```

-Wreorder

C++出现，构造函数中成员变量初始化与声明的顺序不一致。

-Woverflow

范围溢出。

-Wshadow

局部变量覆盖参数、全局变量，报警告

-W 等效于 -Wextra ： 额外的意思