1. External Linkage（外部链接）：

变量或函数具有外部链接时，它们可以在不同的源文件中访问。

通过使用关键字 `extern` 声明或在全局范围定义的变量具有外部链接。

通常，这些变量或函数被放在一个单独的源文件中，并且可以通过其他源文件中的 `extern` 声明来使用。

外部链接的变量或函数在整个程序中都是可见的。

2. Static Linkage（静态链接）：

变量或函数具有静态链接时，它们仅在定义它们的源文件中可见，不能在其他源文件中访问。

使用 `static` 关键字在全局范围定义的变量或函数具有静态链接。

静态链接的变量或函数对于同一源文件中的其他函数来说是全局的，但对其他源文件是不可见的。

3. Namespace（命名空间）：

命名空间是一种用于避免名称冲突的C++特性，允许将变量、函数和类封装在一个特定的命名空间中。

命名空间通过 `namespace` 关键字定义，如 `namespace MyNamespace { /\* ... \*/ }`。

在命名空间内，可以定义全局变量、函数和类，并使用 `::` 操作符在不同的命名空间中访问它们。

命名空间提供了一种组织和隔离代码的方法，以防止名称冲突。·在类声明中声明成员函数时，static关键字将指定类的所有实例共享该函数。 由于函数没有隐式this指针，因此静态成员函数不能访问实例成员。 若要访问实例成员，请使用作为实例指针或引用的参数来声明函数。

·不能将联合成员声明为静态的。但是，全局声明的匿名联合必须是显式声明的static。

编写程序过程中，名称（name）可以是符号常量、变量、函数、结构、枚举、类和对象等等。工程越大，名称互相冲突性的可能性越大。另外使用多个厂商的类库时，也可能导致名称冲突。为了避免，在大规模程序的设计中，以及在程序员使用各种各样的C++库时，这些标识符的命名发生冲突，标准C++引入关键字 namespace（命名空间/名字空间/名称空间），可以更好地控制标识符的作用域。

例如，我们在C语言中，通过static可以限制名字只在当前编译单元内可见，在C++中我们通过namespace来控制对名字的访问。

namespace的语法：

·namespace中可定义常量、变量、函数、结构体、枚举、类等

·namespace只能在全局定义。

·namespace支持嵌套定义。

·namespace是开放的，可随时添加新的成员。

·namespace关键字可以为已有空间名字增加别名

·无名命名空间意味着命名空间中的符号只能在本文件中访问，相当于给符号增加了static修饰。