标准的C/C++工程组织结构，其文件夹可以下图所示的结构来组织。

TestProj ——Include

——Source

——Shared

——Resource

——Debug

——Release

——Bin

——IDE生成的文件

——Readme

——其他临时文件

——数据文件

——配置文件

——DLL

上图中目录的用途如下。

（1）Include目录存放应用程序的头文件（.h），还可以再细分子目录。

（2）Source目录存放应用程序的源文件（.c或 .cpp），还可以再细分子目录。

（3）Shared目录存放一些共享的文件。

（4）Resource目录存放应用程序所用的各种资源文件，包括图片、视频、音频、图标、光标、对话框等，还可以再细分子目录。 （5）Debug目录存放应用程序调试版本生成的中间文件。

（6）Release目录存放应用程序发行版本生成的中间文件。

（7）Bin目录存放程序员自己创建的lib文件和dll文件。

【**注意**】： 分清楚编译时相对路径和运行时相对路径的不同，这在编写操作DLL文件、INI文件及数据文件等外部文件的代码时很重要，因为它们的“参照物”不同。

例如下面的代码行： #include "..\include\abc.h" 是相对于当前工程所在目录的路径，或者是相对于当前文件所在目录的路径，在编译选项的设置中也有这样的路径。

而下面一行代码： OpenFile("..\abc.ini"); 则是相对于运行时可执行文件所在目录的路径，或者是相对于你为当前程序设置的工作目录的路径。

实践总结：

1、不同类型项目的目录结构不尽相同，linux下c语言信息管理系统的一个典型的目录结构如下（不使用IDE开发）：

Campost ——include（头文件目录）

——res（资源文件）

——src（源代码文件）

——lib（库文件）

——bin（编译结果的可执行程序文件）

——config（配置文件）

注：文件名小写简称为好，方便输入；

2、Linux内核源代码的组成如下（假设相对于linux目录）：

include—— 这个目录包括了核心的大多数include文件。另外对于每种支持的体系结构分别有一个子目录。

lib—— 此目录包含了核心的库代码。与处理器结构相关库代码被放在arch/\*/lib/目录下。

arch—— 这个子目录包含了此核心源代码所支持的硬件体系结构相关的核心代码。如对于X86平台就是i386。

init—— 此目录包含核心启动代码。

mm—— 此目录包含了所有的内存管理代码。与具体硬件体系结构相关的内存管理代码位于arch/\*/mm目录下，如对应于X86的就是arch/i386/mm/fault.c

drivers—— 系统中所有的设备驱动都位于此目录中。它又进一步划分成几类设备驱动，每一种也有对应的子目录，如声卡的驱动对应于drivers/sound。

ipc—— 此目录包含了核心的进程间通讯代码。

modules—— 此目录包含已建好可动态加载的模块。

fs—— Linux支持的文件系统代码。不同的文件系统有不同的子目录对应，如ext2文件系统对应的就是ext2子目录。

kernel—— 主要核心代码。同时与处理器结构相关代码都放在arch/\*/kernel目录下。

net—— 核心的网络部分代码。里面的每个子目录对应于网络的一个方面。

scripts—— 此目录包含用于配置核心的脚本文件。

Documentation—— 此目录是一些文档，起参考作用。

参考文献:

[1]. 参考：《高质量程序设计指南C++/C语言》第三版。