Cuint 实践入门

A Practice Lecture on CUnit

目录

一、引言

写在引言里的废话：

本文的成文参考了众多的网页、论文与书籍等资料，按照惯例，将在章节的后面附上引用文献的来源以及参考资料目录。如有遗漏或不正当地使用了某种资料，请致信 openlek@live.cn，作者将及时更新本文档，并对相关内容做处理。

Let’s GO!

二、基础知识——为什么要单元测试

我们知道，在开发时越早发现 BUG，就能节省更多的时间，降低更多的风险。单元测试是在软件开发过程中要进行的最低级别的测试活动，在单元测试活动中，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。在传统的结构化编程语言中，比如 C，要进行测试的单元一般是函数或子过程。在象C++这样的面向对象的语言中， 要进行测试的基本单元是类。

经验表明一个尽责的单元测试方法将会在软件开发的某个阶段发现很多的 Bug，并且修改它们的成本也很低。在软件开发的后期阶段，Bug 的发现并修改将会变得更加困难，并要消耗大量的时间和开发费用。无论什么时候作出修改都要进行完整的回归测试，在生命周期中尽早地对软件产品进行测试将使效率和质量得到最好的保证。 在提供了经过测试的单元的情况下，系统集成过程将会大大地简化。开发人员可以将精力集中在单元之间的交互作用和全局的功能实现上，而不是陷入充满很多 Bug 的单元之中不能自拔。

其实我们每天都在做单元测试。你写了一个函数，除了极简单的外，总是要执行一下，看看功能是否正常，有时还要想办法输出些数据，如弹出信息窗口什么的，这也是单元测试，可以把这种单元测试称为临时单元测试。只进行了临时单元测试的软件，针对代码的测试很不完整，代码覆盖率要超过 70%都很困难，未覆盖的代码可能遗留大量的细小的错误，这些错误还会互相影响，当 BUG 暴露出来的时候难于调试，大幅度提高后期测试和维护成本，也降低了开发商的竞争力。可以说，进行充分的单元测试，是提高软件质量，降低开发成本的必由之路。

最后，单元测试不仅仅是作为无错编码一种辅助手段在一次性的开发过程中使用，单元测试必须是可重复的，无论是在软件修改，或是移植到新的运行环境的过程中。因此，所有的测试都必须在整个软件系统的生命周期中进行维护。

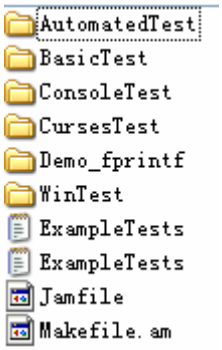
三、编译第一个测试程序

1、编译 CUnit.lib 文件

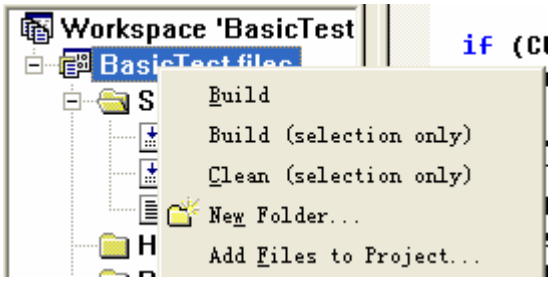
在 CUnit 目录下，打开 CUnit.dsp，编译即可得到 CUnit.lib 文件。

2、运行 BasicTest 示例

Examples 目录如下，包含了四种测试模式的示例，ExampleTests.c 是单元测试程序，所有测试示例都使用它。



将 ExampleTests.c 与 ExampleTests.h 拷贝到 BasicTest 目录下，并添加到工程中，编译后即可运行看结果。

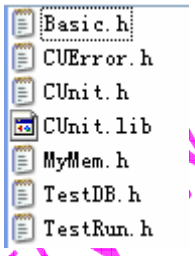


3、如果编译时提示没有 ExampleTests.h，注意在工程中把这 2 个文件包含近来。如果提示没有 CUnit.lib 文件，则按照第一步先编译 CUnit.dsp。

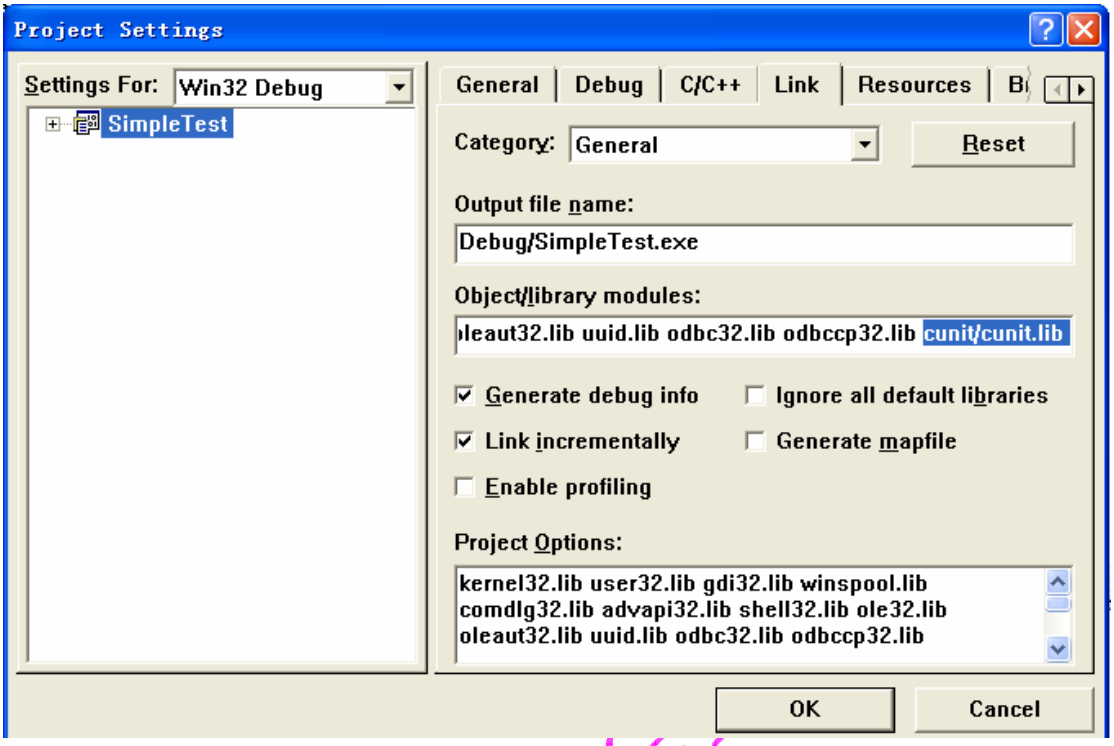
四、编写第一个测试程序

http://cunit.sourceforge.net/example.html 网站上提供了一个简单入门的 demo。针对 C 语言标准库函数 fprintf() and fread()进行黑盒单元测试。

1、。建立 SimpleTest.c 文件保存网页中的代码，建立文件夹 Cunit 保存 Cunit.lib 和相关的头文件。相关代码在 Examples/SimpleTest 目录下



2、设置编译选项，主要是链接 lib 文件，编译后即可运行看结果。



3、代码阅读。由于测试任务是对函数 fprintf() & fread()进行黑盒单元测试，因此测试用例设计如下：

fprintf	测试输入	测试输出
用例 1	输入一个空字符串	返回 0
用例 2	输入一个带换行符的字符串	返回字符串长度
用例 3	输入一个带不定参数的字符串	返回字符串的实际长度

fread	测试输入	测试输出
用例 1	定义一个数组读取文件	返回实际读取长度值，数组内容正确

代码实现原理：打开一个临时文件，测试函数使用 fprintf() & fread()读写该临时文件，然后检查函数返回结果。

细节讲解：CUnit 相关函数原形如下，

CU_pSuite CU_add_suite(const char* strName, CU_InitializeFunc pInit, CU_CleanupFunc pClean)

CU_pTest CU_add_test(CU_pSuite pSuite, const char* strName, CU_TestFunc pTestFunc)

main 函数中，蓝色标记的函数完成打开&关闭文件的操作。红色标记的函数实现了上述 4 个测试用例。

```
int main()
{
    CU_pSuite pSuite = NULL;

    /* initialize the CUnit test registry */
```

```
    if (CUE_SUCCESS != CU_initialize_registry())
        return CU_get_error();

    /* add a suite to the registry */
    pSuite = CU_add_suite("Suite_1", init_suite1, clean_suite1);
    if (NULL == pSuite) {
        CU_cleanup_registry();
        return CU_get_error();
    }

    /* add the tests to the suite */
    /* NOTE - ORDER IS IMPORTANT - MUST TEST fread() AFTER fprintf() */
    if ((NULL == CU_add_test(pSuite, "test of fprintf()", testFPRINTF)) ||
        (NULL == CU_add_test(pSuite, "test of fread()", testFREAD)))
    {
        CU_cleanup_registry();
        return CU_get_error();
    }

    /* Run all tests using the CUnit Basic interface */
    CU_basic_set_mode(CU_BRM_VERBOSE);
    CU_basic_run_tests();
    CU_cleanup_registry();
    return CU_get_error();
}
```

五、后记

作　者：
Email: openlek@live.cn
http://www.openlek.com
版本历史
2009.08.05　初稿

版权声明

（2）未经作者许可，不得公开出版或大量印发以获取利益，以传播知识为目的电子拷贝除外。

（3）请遵守其他法律规定，以免发生纠纷。

欢迎读者对本文档提出批评建议，信息反馈请联系 Email：openlek@live.cn。

Copyright