

软件测试

杰能二部

目录

一、软件测试-通用知识

二、软件测试-工具链

三、软件测试具体相关

软件测试-通用知识

通用知识



软件测试定义

软件测试分类

白盒测试

通用知识 软件测试定义

使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。

- 1983年IEEE提出的软件工程术语

测试是以评价一个程序或者系统属性为目标的任何一种活动。测试是对软件质量的度量。

- Bill Hetzel在《软件测试完全指南》

通用知识 软件测试定义 软件的质量

软件质量就是“软件与明确地和隐含地定义的需求相一致的程度”。更具体地说，软件质量是软件与明确地叙述的功能和性能需求、文档中明确描述的开发标准以及任何专业开发的软件产品都应该具有的隐含特征相一致的程度。

从管理角度对软件质量进行度量，可将影响软件质量的主要因素划分为三组：

产品运行（可用性、性能、安全性、易用性）；

产品修改（可修改性、可测试性）；

产品转移（可移植性、互运行性）；

可用性：系统能够正常运行的时间比例。（硬件损坏，工作环境恶劣）

可修改性：指能够快速地对系统性能价格比进行变更的能力。（CAN程序更新，远程更新程序）

性能：指系统的响应能力，即经过多长时间才能对某个事件作出响应，或者在某个时间内系统所能处理事件的个数。（转矩响应速度）

安全性：衡量系统在向合法用户提供服务的同时，阻止非授权使用的能力。

可测试性：指软件发现故障并隔离，定位其故障的能力特性，以及在一定时间和成本前提下，进行测试设计，测试执行的能力。

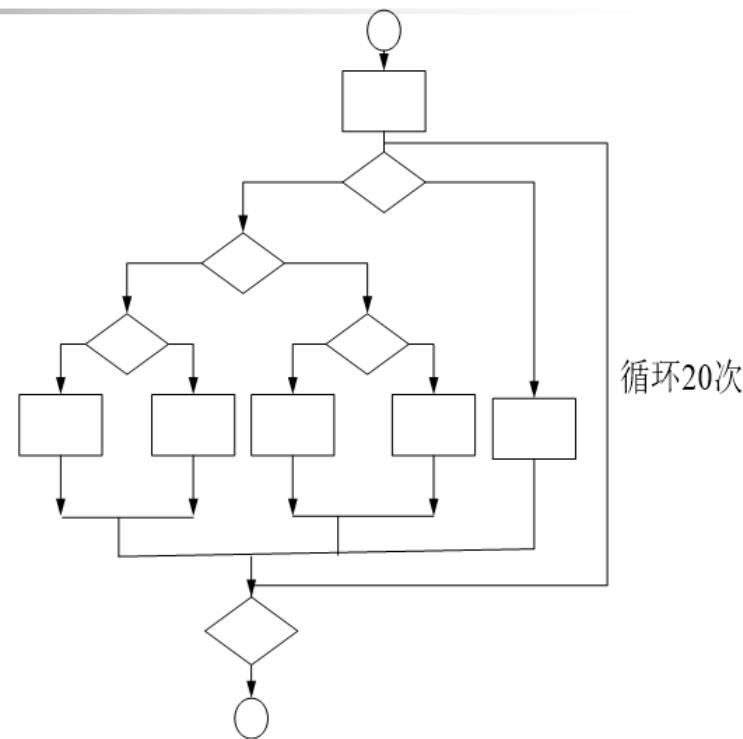
易用性：衡量用户使用一个软件产品完成指定任务的难易程度。

通用知识 软件测试定义

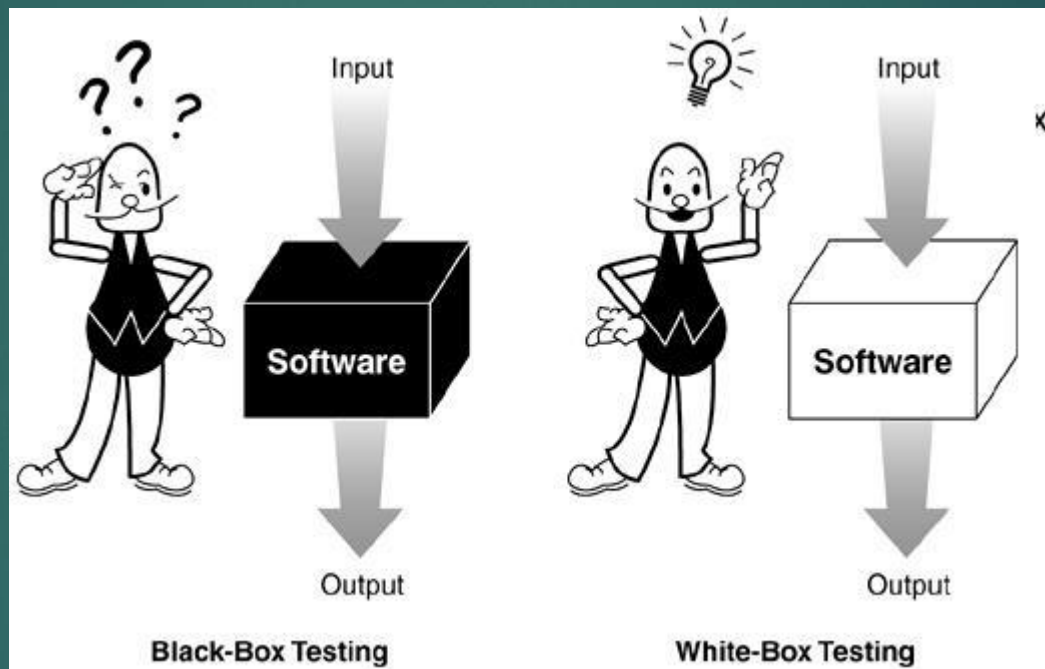
为什么需要设计软件测试

典型的群举测试

这个流程图，其中包括了一个执行达**20**次的循环。那么它所包含的不同执行路径数高达 **5^{20}** 条，若要对它进行穷举测试，覆盖所有的路径。假使测试程序对每一条路径进行测试需要**1**毫秒，同样假定一天工作**24**小时，一年工作**365**天，那么要想把如图所示的小程序的所有路径测试完，则需要**3170**年。

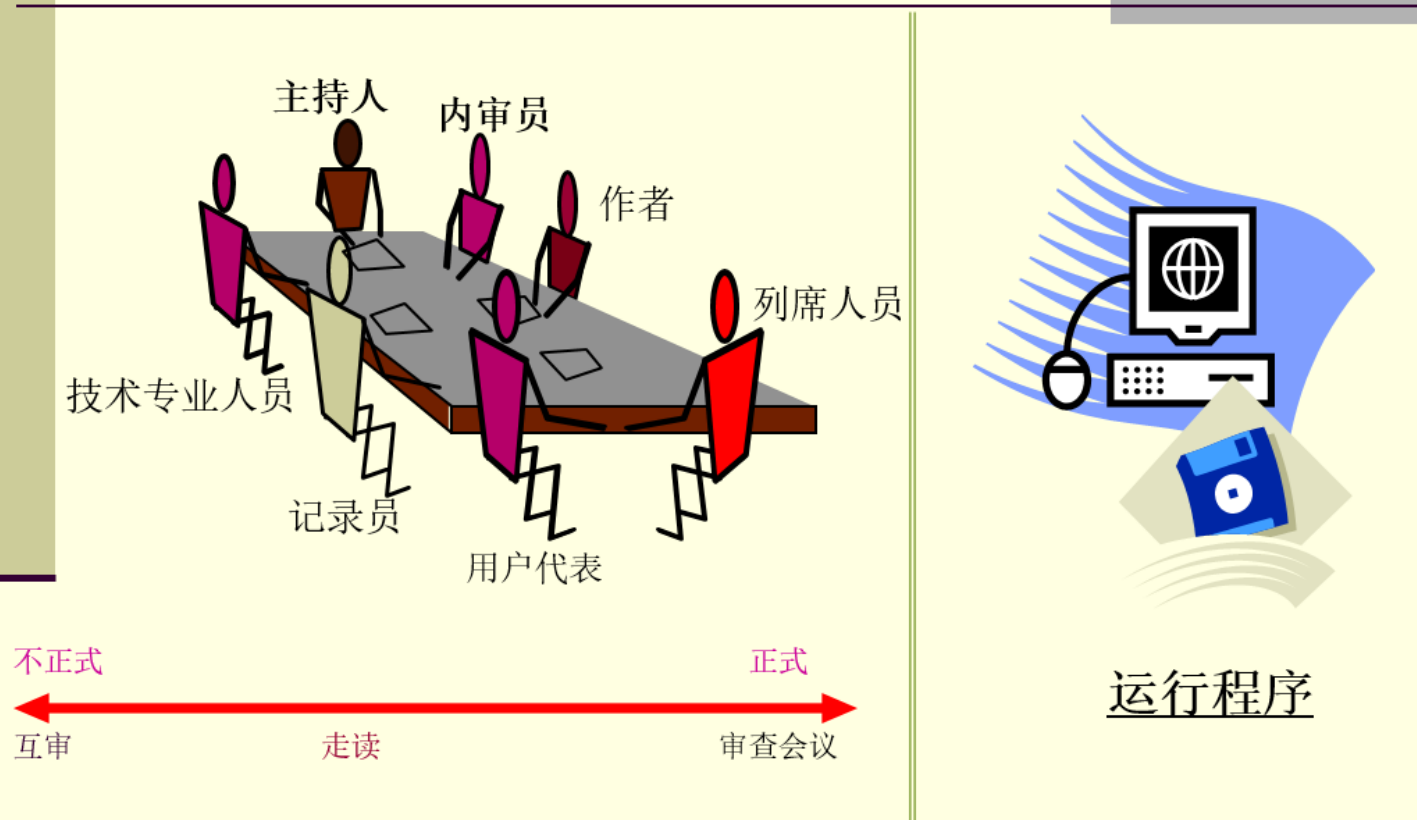


通用知识 软件测试分类



通用知识 软件测试分类

静态的和动态的



通用知识 软件测试分类

自动测试和手工测试



手工模拟用户
操作

通用知识 白盒测试

语句覆盖

判定覆盖

条件覆盖

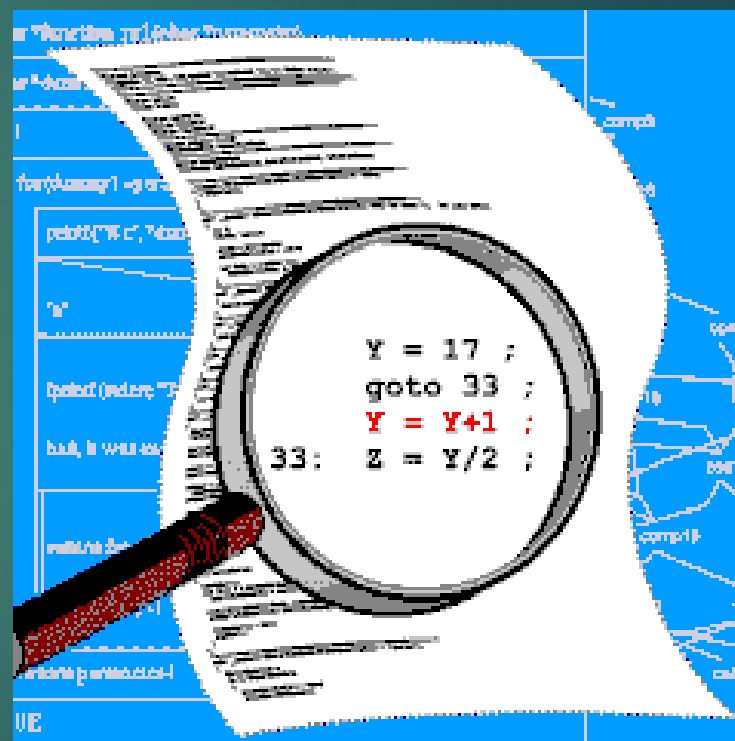
判定条件覆盖

条件组合覆盖

路径覆盖

基本路径测试法

静态白盒测试法



通用知识 白盒测试 单元测试

单元测试是开发者编写的一小段代码，用于检验被测代码的一个很小的、很明确的功能是否正确。通常而言，一个单元测试是用于判断某个特定条件（或者场景）下某个特定函数的行为。

单元测试是由程序员自己来完成，最终受益的也是程序员自己。

测试一个模块时，可能需要为该模块编写一个驱动模块和若干个桩模块。驱动模块用来调用被测模块，它接收测试者提供的测试数据，并把这些数据传送给被测模块，然后从被测模块接收测试结果，并以某种可见的方式将测试结果返回给测试人员；桩模块用来模拟被测模块所调用的子模块，它接受被测模块的调用，检验调用参数，并以尽可能简单的操作模拟被调用的子程序模块功能，把结果送回被测模块。

通用知识 白盒测试 代码覆盖

在做单元测试时，代码覆盖率常常被拿来作为衡量测试好坏的指标，甚至，用代码覆盖率来考核测试任务完成情况，比如，代码覆盖率必须达到80%或 90%。于是乎，测试人员费尽心思设计案例覆盖代码。

- 覆盖率数据只能代表你测试过哪些代码，不能代表你是否测试好这些代码。
- 不要只拿语句覆盖率(行覆盖率)来考核你的测试人员。
- 路径覆盖率 > 判定覆盖 > 语句覆盖
- 测试人员不能盲目追求代码覆盖率，而应该想办法设计更多更好的案例，哪怕多设计出来的案例对覆盖率一点影响也没有。

通用知识 白盒测试 走查与审查

| 项目 | 走 查 | 审 查 |
|--------|--------------|---|
| 准备 | 通读设计和编码 | 应准备好需求描述文档、程序设计文档、程序的源代码清单、代码编码标准和代码缺陷检查表 |
| 形式 | 非正式会议 | 正式会议 |
| 参加人员 | 开发人员为主 | 项目组成员包括测试人员 |
| 主要技术方法 | 无 | 缺陷检查表 |
| 注意事项 | 限时、不要现场修改代码 | 限时、不要现场修改代码 |
| 生成文档 | 会议记录 | 静态分析错误报告 |
| 目标 | 代码标准规范，无逻辑错误 | 代码标准规范，无逻辑错误 |

工具链 静态检查PC-LINT9.0+ Source Insight 3.5

- PC-Lint是Gimpel Software公司开发的一个C/C++静态语法检查工具。一般来说，软件在编译连接通过后就可以使用PC-Lint做静态检查。
- Source Insight是一个面向项目开发的程序编辑器和代码浏览器，它拥有内置的对C/C++, C#和Java等程序的分析。能分析源代码并在工作的同时动态维护它自己的符号数据库，并自动显示有用的上下文信息。

工具链 静态检查PC-LINT9.0+ Source Insight 3.5

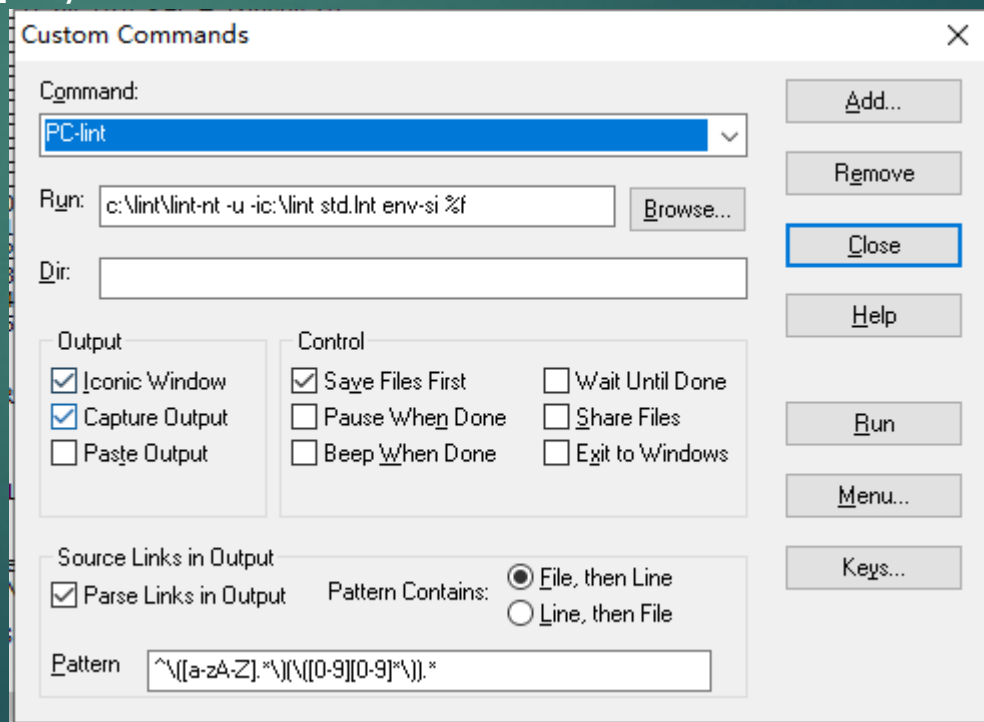
```
// Texas Inst. TI320 class C compilers, , lib-gtk.lnt lib-stl.lnt
// Standard lint options

au-misra1-ccj.lnt
co-ti320.lnt
lib-ccj.lnt
options.lnt

-iD:\si_test\PSFB_PCMC
-iD:\si_test\PSFB_PCMC\C2000_Library\ti\controlSUITE\development_kits\~SupportFiles\F2803x_headers
-iD:\si_test\PSFB_PCMC\C2000_Library\ti\controlSUITE\device_support\f2803x\v127\DSP2803x_common\include
-iD:\si_test\PSFB_PCMC\C2000_Library\ti\controlSUITE\device_support\f2803x\v127\DSP2803x_headers\include
-iD:\si_test\PSFB_PCMC\C2000_Library\ti\controlSUITE\libs\app_libs\digital_power\f2803x_v3.2\include
-iD:\si_test\PSFB_PCMC\C2000_Library\ti\controlSUITE\libs\math\IQmath\v160\include
-iD:\si_test\PSFB_PCMC\base
-iD:\si_test\PSFB_PCMC\M_lib
```


工具链 静态检查PC-LINT9.0+ Source Insight 3.5

- 1) 从Options菜单中选择“Custom Commands”命令项。
- 2) 在Name栏中输入“PC-lint”，这个名称可以随便起，只要清晰即可。
- 3) 在Run栏中输入“c:/lint/lint-nt -u -ic:/lint std env-si %f”其中c:/lint是你PC-LINT的安装目录。
- 4) 在Output栏中选择“Iconic Window”、“Capture Output”。
- 5) 在Control栏中选择“Save Files First”。
- 6) 在Source Links in Output栏中选择“Parse Links in Output”、“File, then Line”。
- 7) 在Pattern栏中输入“^\\([a-zA-Z]*\\)\\/([0-9]+\\)”。
- 8) 点Add键加入该命令



工具链 静态检查PC-LINT9.0+ Source Insight 3.5

```
PC-lint for C/C++ (NT) Vers. 9.00a, Copyright Gimpel Software 1985-2008

--- Module:   D:\si_test\PSFB_PCMC\M_lib\ADC_SOC.c (C)

typedef interrupt void(*PINT)(void);
D:\si_test\PSFB_PCMC\C2000_Library\ti\controlSUITE\device_support\f2803x\v127\DSP2803x_headers\include\DSP2803x_PieVect.h
 24 Error 140: Type appears after modifier, use '+fem'

void ADC_SOC_CNF(int ChSel[], int TrigsSel[], int ACQPS[], int IntChSel, int mode) {
D:\si_test\PSFB_PCMC\M_lib\ADC_SOC.c 39 Note 957: Function 'ADC_SOC_CNF'
  defined without a prototype in scope [MISRA Rule 71]

#... sm(" EALLOW")
  EALLOW;
D:\si_test\PSFB_PCMC\M_lib\ADC_SOC.c 42 Error 10: Expecting identifier or other
  declarator

#... LLOW")
  EALLOW;
D:\si_test\PSFB_PCMC\M_lib\ADC_SOC.c 42 Error 10: Expecting ')'

  EALLOW;
D:\si_test\PSFB_PCMC\M_lib\ADC_SOC.c 42 Error 2: Unclosed Quote

  DSP28x_usDelay(1000); // Delay before converting ADC channels
D:\si_test\PSFB_PCMC\M_lib\ADC_SOC.c 49 Note 917: Prototype coercion (arg. no.
  1) int to unsigned long [MISRA Rule 77]

#... asm(" EDIS")
  EDIS;
D:\si_test\PSFB_PCMC\M_lib\ADC_SOC.c 130 Error 42: Expected a statement

--- Global Wrap-up

0 Note 974: Worst case function for stack usage: 'ADC_SOC_CNF' is finite,
  requires 45 bytes total stack in calling 'DSP28x_usDelay' (external). See
  +stack for a full report. [MISRA Rule 70]
```

```
00024: typedef interrupt void(*PINT)(void);
```

不识别关键字interrupt

17.6.48 —957— 没有在范围内的原型的函数 '**Symbol**' 定义

对所有的函数在头文件内声明原型通常是一个很好的实践，格

```
: #define EALLOW asm(" EALLOW")
: #define EDIS asm(" EDIS")
```

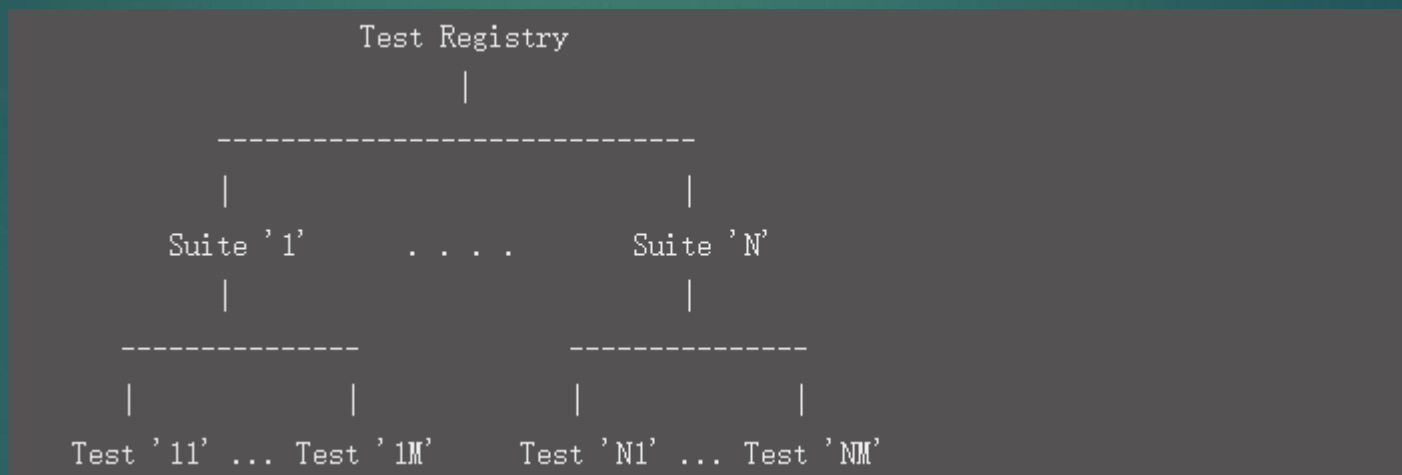
不识别关键字“EALLOW”

17.6.10 —917—Type 到 Type 强制原型转换

注意： 无论何时，一个隐含的数学转换的发生是一个原型转换的结果。

工具链 白盒测试CUnit +Lcov

- CUnit是一种C语言单元测试框架，继JUnit CppUnit的成功后，c语言环境下也出现了开放源码的白盒测试用例CUnit。CUnit以静态库的形式提供给用户使用，用户编写程序的时候直接链接此静态库就可以了。它提供了一个简单的单元测试框架，并且为常用的数据类型提供了丰富的断言语句支持。



- Lcov是GCOV图形化的前端工具是Linux Test Project维护的开放源代码工具，最初被设计用来支持Linux内核覆盖率的度量基于Html输出，并生成一棵完整的HTML树输出包括概述、覆盖率百分比、图表，能快速浏览覆盖率数据支持大项目，提供三个级别的视图：目录视图、文件视图、源码视图。

工具链 白盒测试CUnit +Lcov

安装CUnit

```
0. 获取root
sudo -s
1. 安装automake
apt-get install autoconf
2. 安装libtool
apt-get install aptitude
aptitude install libtool
```

- 一、CUnit-2.1-3: 下载地址<https://sourceforge.net/projects/cunit/>
- 二、解压: #tar jxvf CUnit-2.1-3.tar.bz2
- 三、安装: 进入CUnit-2.1-3目录, 执行以下命令:

```
autoscan
```

```
mv configure.in configure.ac
```

```
aclocal
```

```
autoheader
```

```
libtoolize --automake --copy --debug --force
```

```
automake
```

```
autoconf
```

```
./configure 【./configure --prefix <Your choice of directory for installation> example: ./configure --prefix  
/usr/unittest】
```

```
make
```
























```
sudo make install
```

工具链 例子 源文件

```
00037: #include "PeripheralHeaderIncludes.h"
00038:
00039: void ADC_SOC_CNF(int ChSel[], int Trigsel[], int ACQPS[], int IntChSel, int mode) {
00040:     extern void DSP28x_usDelay(uint32 Count);
00041:
00042:     EALLOW;
00043:     AdcRegs.ADCCTL1.bit.ADCREFSEL = 0;
00044:     AdcRegs.ADCCTL1.bit.ADCBGPWD = 1;           // Power up band gap
00045:     AdcRegs.ADCCTL1.bit.ADCREFPWD = 1;         // Power up reference
00046:     AdcRegs.ADCCTL1.bit.ADCPWDN = 1;           // Power up rest of ADC
00047:     AdcRegs.ADCCTL1.bit.ADCENABLE = 1;         // Enable ADC
00048:
00049:     DSP28x_usDelay(1000);                       // Delay before converting ADC channels
00050:
00051:     AdcRegs.ADCCTL1.bit.INTPULSEPOS = 1;
00052:
00053:     AdcRegs.ADCSOC0CTL.bit.ACQPS = ACQPS[0];
00054:     AdcRegs.ADCSOC1CTL.bit.ACQPS = ACQPS[1];
00055:     AdcRegs.ADCSOC2CTL.bit.ACQPS = ACQPS[2];
00056:     AdcRegs.ADCSOC3CTL.bit.ACQPS = ACQPS[3];
00057:     AdcRegs.ADCSOC4CTL.bit.ACQPS = ACQPS[4];
00058:     AdcRegs.ADCSOC5CTL.bit.ACQPS = ACQPS[5];
00059:     AdcRegs.ADCSOC6CTL.bit.ACQPS = ACQPS[6];
00060:     AdcRegs.ADCSOC7CTL.bit.ACQPS = ACQPS[7];
00061:     AdcRegs.ADCSOC8CTL.bit.ACQPS = ACQPS[8];
00062:     AdcRegs.ADCSOC9CTL.bit.ACQPS = ACQPS[9];
00063:     AdcRegs.ADCSOC10CTL.bit.ACQPS = ACQPS[10];
00064:     AdcRegs.ADCSOC11CTL.bit.ACQPS = ACQPS[11];
00065:     AdcRegs.ADCSOC12CTL.bit.ACQPS = ACQPS[12];
00066:     AdcRegs.ADCSOC13CTL.bit.ACQPS = ACQPS[13];
00067:     AdcRegs.ADCSOC14CTL.bit.ACQPS = ACQPS[14];
00068:     AdcRegs.ADCSOC15CTL.bit.ACQPS = ACQPS[15];
00069:
00070:     AdcRegs.INTSEL1N2.bit.INT1SEL = IntChSel;    // IntChSel causes ADCInterrupt 1
00071:
00072:     if (mode == 0){ // Start-Stop conversion mode
00073:         AdcRegs.ADCINTFLG.bit.ADCINT1 = 0;       // clear interrupt flag for ADCINT1
00074:         AdcRegs.INTSEL1N2.bit.INT1CONT = 0;      // clear ADCINT1 flag to begin a new set of conversions
00075:         AdcRegs.ADCINTSOCSEL1.all = 0x0000;      // No ADCInterrupt will trigger SOCx
00076:         AdcRegs.ADCINTSOCSEL2.all = 0x0000;
00077:     }
00078:     if (mode == 1){ // Continuous conversion mode
00079:         AdcRegs.INTSEL1N2.bit.INT1CONT = 1;      // set ADCInterrupt 1 to auto clr
00080:         AdcRegs.ADCINTSOCSEL1.all = 0xFF;        // ADCInterrupt 1 will trigger SOCx, TrigSel is ignored
00081:         AdcRegs.ADCINTSOCSEL2.all = 0xFF;
00082:     }
00083:
00084:     if (mode == 2){ // CLA mode, Start Stop ADC with auto clr ADC Flag
00085:         AdcRegs.ADCINTFLG.bit.ADCINT1 = 0;       // clear interrupt flag for ADCINT1
00086:         AdcRegs.INTSEL1N2.bit.INT1CONT = 1;      // set ADCInterrupt 1 to auto clr
00087:         AdcRegs.ADCINTSOCSEL1.all = 0x0000;      // No ADCInterrupt will trigger SOCx
00088:         AdcRegs.ADCINTSOCSEL2.all = 0x0000;
00089:     }
00090:
00091:     if (IntChSel < 16)
00092:         AdcRegs.INTSEL1N2.bit.INT1E = 1;         // enable ADC interrupt 1
00093:     else
00094:         AdcRegs.INTSEL1N2.bit.INT1E = 0;         // disable the ADC interrupt 1
00095:
00096:     // Select the channel to be converted when SOCx is received
00097:     AdcRegs.ADCSOC0CTL.bit.CHSEL = ChSel[0];
00098:     AdcRegs.ADCSOC1CTL.bit.CHSEL = ChSel[1];
00099:     AdcRegs.ADCSOC2CTL.bit.CHSEL = ChSel[2];
00100:     AdcRegs.ADCSOC3CTL.bit.CHSEL = ChSel[3];
00101:     AdcRegs.ADCSOC4CTL.bit.CHSEL = ChSel[4];
00102:     AdcRegs.ADCSOC5CTL.bit.CHSEL = ChSel[5];
```


工具链 例子

导入库文件 并修改

| | | | |
|---|-----------------|------|-------|
|  DSP2803x_Adc.h | 2019/7/23 13:22 | H 文件 | 17 KB |
|  DSP2803x_BootVars.h | 2019/7/23 13:22 | H 文件 | 2 KB |
|  DSP2803x_Cla.h | 2019/7/23 13:22 | H 文件 | 8 KB |
|  DSP2803x_Comp.h | 2019/7/23 13:22 | H 文件 | 4 KB |
|  DSP2803x_CpuTimers.h | 2019/7/23 13:22 | H 文件 | 5 KB |
|  DSP2803x_DevEmu.h | 2019/7/23 13:22 | H 文件 | 3 KB |
|  DSP2803x_Device.h | 2019/7/23 13:22 | H 文件 | 7 KB |
|  DSP2803x_ECan.h | 2019/7/23 13:22 | H 文件 | 45 KB |
|  DSP2803x_ECap.h | 2019/7/23 13:22 | H 文件 | 6 KB |
|  DSP2803x_EPwm.h | 2019/7/23 13:22 | H 文件 | 23 KB |
|  DSP2803x_EQep.h | 2019/7/23 13:22 | H 文件 | 10 KB |
|  DSP2803x_Gpio.h | 2019/7/23 13:22 | H 文件 | 13 KB |
|  DSP2803x_HRCap.h | 2019/7/23 13:22 | H 文件 | 7 KB |
|  DSP2803x_I2c.h | 2019/7/23 13:22 | H 文件 | 8 KB |
|  DSP2803x_Lin.h | 2019/7/23 13:22 | H 文件 | 23 KB |
|  DSP2803x_NmiIntrupt.h | 2019/7/23 13:22 | H 文件 | 4 KB |
|  DSP2803x_PieCtrl.h | 2019/7/23 13:22 | H 文件 | 6 KB |
|  DSP2803x_PieVect.h | 2019/7/29 17:20 | H 文件 | 7 KB |
|  DSP2803x_Sci.h | 2019/7/23 13:22 | H 文件 | 8 KB |
|  DSP2803x_Spi.h | 2019/7/23 13:22 | H 文件 | 7 KB |
|  DSP2803x_SysCtrl.h | 2019/7/23 13:22 | H 文件 | 17 KB |
|  DSP2803x_XIntrupt.h | 2019/7/23 13:22 | H 文件 | 2 KB |
|  PeripheralHeaderIncludes.h | 2019/7/29 17:21 | H 文件 | 5 KB |

PeripheralHeaderIncludes.h

```
00045: //CCJ #define EALLOW asm(" EALLOW")
00046: #define EALLOW
00047: //CCJ #define EDIS asm(" EDIS")
00048: #define EDIS
```

DSP2803x_PieVect.h

```
00024: //CCJ typedef interrupt void(*PINT)(void);
00025: typedef void(*PINT)(void);
```

工具链 例子

编写桩文件和测试文件

桩文件p.c

```
p.c X
(Unknown Scope)
#include "PeripheralHeaderIncludes.h"
#include <stdio.h>
volatile struct ADC_REGS AdcRegs;
void DSP28x_usDelay(Uint32 Count)
{
    printf("DSP28x_usDelay \n");
}
```

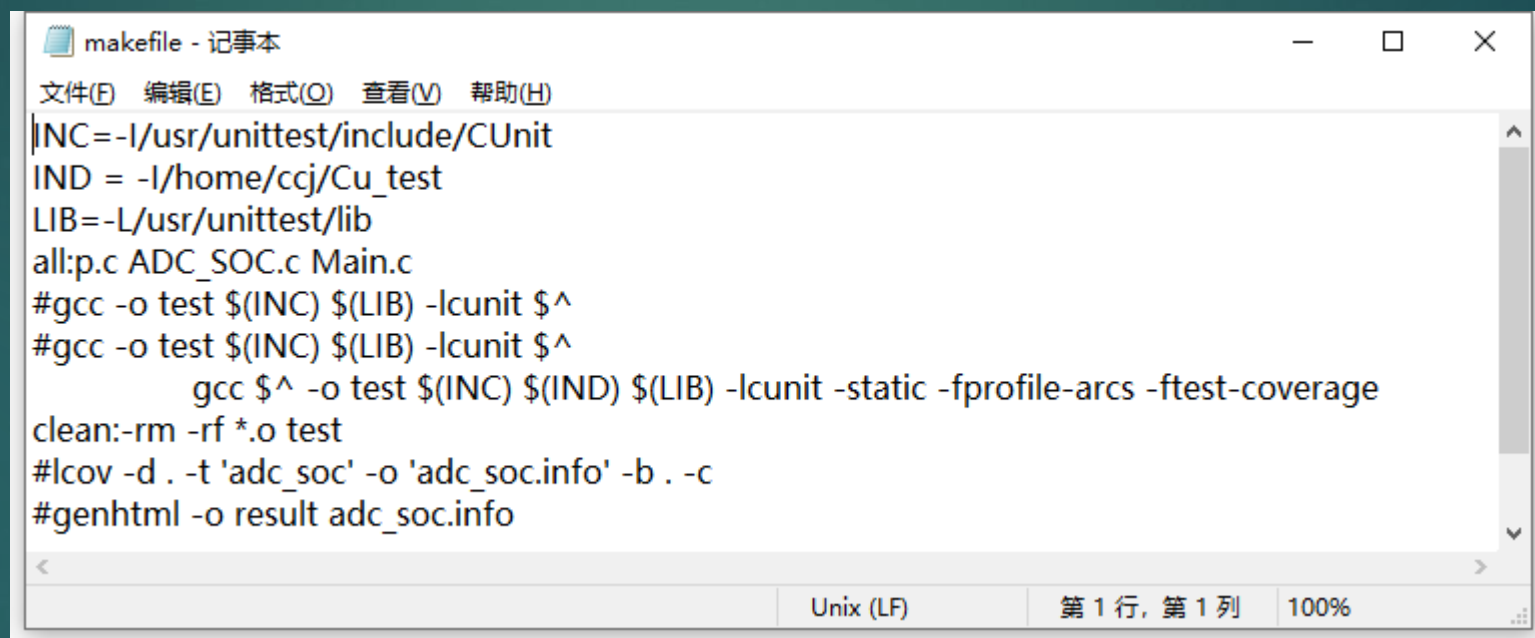
```
00001: #include <stdio.h>
00002: #include <stdlib.h>
00003: #include <string.h>
00004: #include <assert.h>
00005: #include "Basic.h"
00006: #include "PeripheralHeaderIncludes.h"
00007:
00008: /**/ *--- functions to be tested ----- */
00009: extern void ADC_SOC_CNF(int ChSel[], int TrigsSel[], int ACQPS[], int IntChSel, int mode);
00010: int chs[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
00011: int ts[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
00012: int acs[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
00013: int ic= 15;
00014: int cmode =0;
00015: /* Pointer to the file used by the tests. */
00016: static FILE* temp_file = NULL;
00017:
00018: /* The suite initialization function.
00019:  * Opens the temporary file used by the tests.
00020:  * Returns zero on success, non-zero otherwise.
00021:  */
00022: int init_suite1(void)
00023: {
00024:     if (NULL == (temp_file = fopen("temp.txt", "w+"))) {
00025:         return -1;
00026:     }
00027:     else {
00028:         return 0;
00029:     }
00030: }
00031:
00032: /* The suite cleanup function.
00033:  * Closes the temporary file used by the tests.
00034:  * Returns zero on success, non-zero otherwise.
00035:  */
00036: int clean_suite1(void)
00037: {
00038:     if (0 != fclose(temp_file)) {
00039:         return -1;
00040:     }
00041:     else {
00042:         temp_file = NULL;
00043:         return 0;
00044:     }
00045: }
00046:
```


工具链 例子 测试文件

```
00047: /**/*----- test cases -----*/
00048: void testmode1()
00049: {
00050:     cmode = 0;
00051:     ic = 1;
00052:     ADC_SOC_CNF(chs,ts,acs,ic,cmode);
00053:     CU_ASSERT_EQUAL(AdcRegs.ADCINTFLG.bit.ADCINT1,0);
00054:     CU_ASSERT_EQUAL(AdcRegs.INTSEL1N2.bit.INT1E,1);
00055:     ic = 16;
00056:     ADC_SOC_CNF(chs,ts,acs,ic,cmode);
00057:     CU_ASSERT_EQUAL(AdcRegs.INTSEL1N2.bit.INT1E,0);
00058: }
00059: void testmode2()
00060: {
00061:     cmode = 1;
00062:     ic = 1;
00063:     ADC_SOC_CNF(chs,ts,acs,ic,cmode);
00064:     CU_ASSERT_EQUAL(AdcRegs.ADCINTFLG.bit.ADCINT1,0);
00065:     CU_ASSERT_EQUAL(AdcRegs.INTSEL1N2.bit.INT1E,1);
00066: }
00067: void testmode3()
00068: {
00069:     cmode = 2;
00070:     ic = 1;
00071:     ADC_SOC_CNF(chs,ts,acs,ic,cmode);
00072:     CU_ASSERT_EQUAL(AdcRegs.ADCINTFLG.bit.ADCINT1,0);
00073:     CU_ASSERT_EQUAL(AdcRegs.INTSEL1N2.bit.INT1E,1);
00074: }
00075: void testmode4()
00076: {
00077:     cmode = 3;
00078:     ic = 1;
00079:     ADC_SOC_CNF(chs,ts,acs,ic,cmode);
00080:     CU_ASSERT_EQUAL(AdcRegs.ADCINTFLG.bit.ADCINT1,0);
00081:     CU_ASSERT_EQUAL(AdcRegs.INTSEL1N2.bit.INT1E,1);
00082: }
00083:
```

```
00084: int main()
00085: {
00086:     CU_pSuite pSuite = NULL;
00087:
00088:     /* initialize the CUnit test registry */
00089:     if (CUE_SUCCESS != CU_initialize_registry())
00090:         return CU_get_error();
00091:     /* add a suite to the registry */
00092:     pSuite = CU_add_suite("Suite_1", init_suite1, clean_suite1);
00093:     if (NULL == pSuite) {
00094:         CU_cleanup_registry();
00095:         return CU_get_error();
00096:     }
00097:
00098:     /* add the tests to the suite */
00099:     /* NOTE - ORDER IS IMPORTANT - MUST TEST fread() AFTER fprintf() */
00100:     if ((NULL == CU_add_test(pSuite, "testmode1", testmode1)) ||
00101:         (NULL == CU_add_test(pSuite, "testmode2", testmode2)) ||
00102:         (NULL == CU_add_test(pSuite, "testmode3", testmode3)) ||
00103:         (NULL == CU_add_test(pSuite, "testmode4", testmode4)))
00104:     {
00105:         CU_cleanup_registry();
00106:         return CU_get_error();
00107:     }
00108:
00109:
00110:
00111:     /* Run all tests using the CUnit Basic interface */
00112:     CU_basic_set_mode(CU_BRM_VERBOSE);
00113:     CU_basic_run_tests();
00114:     CU_cleanup_registry();
00115:     return CU_get_error();
00116: } ? end main ?
```

工具链 例子 makefile



```
makefile - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
INC=-I/usr/unittest/include/CUnit
IND = -I/home/ccj/Cu_test
LIB=-L/usr/unittest/lib
all:p.c ADC_SOC.c Main.c
#gcc -o test $(INC) $(LIB) -lcunit $^
#gcc -o test $(INC) $(LIB) -lcunit $^
        gcc $^ -o test $(INC) $(IND) $(LIB) -lcunit -static -fprofile-arcs -ftest-coverage
clean:-rm -rf *.o test
#lcov -d . -t 'adc_soc' -o 'adc_soc.info' -b . -c
#genhtml -o result adc_soc.info
```

Unix (LF) 第 1 行, 第 1 列 100%

工具链 例子 运行结果

```
ccj@ccj-OptiPlex-745: ~/Cu_test

CUnit - A unit testing framework for C - Version 2.1-3
http://cunit.sourceforge.net/

Suite: Suite_1
  Test: testmode1 ...DSP28x_usDelay
DSP28x_usDelay
passed
  Test: testmode2 ...DSP28x_usDelay
passed
  Test: testmode3 ...DSP28x_usDelay
passed
  Test: testmode4 ...DSP28x_usDelay
passed

Run Summary:   Type   Total    Ran  Passed  Failed  Inactive
               suites    1      1    n/a     0       0
               tests     4      4     4       0       0
               asserts    9      9     9       0     n/a

Elapsed time =   0.000 seconds
ccj@ccj-OptiPlex-745:~/Cu_test$
```

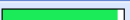
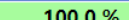


LCOV - code coverage report

Current view: [top level](#)

Test: [adc_soc.info](#)

Date: 2019-07-29

| | Hit | Total | Coverage |
|------------|-----|-------|----------|
| Lines: | 130 | 137 | 94.9 % |
| Functions: | 9 | 9 | 100.0 % |
| Branches: | 16 | 24 | 66.7 % |

| Directory | Line Coverage  | Functions  | Branches  |
|-----------------------------|--|---|--|
| ccj/Cu_test |  94.9 % 130 / 137 | 100.0 % 9 / 9 | 66.7 % 16 / 24 |

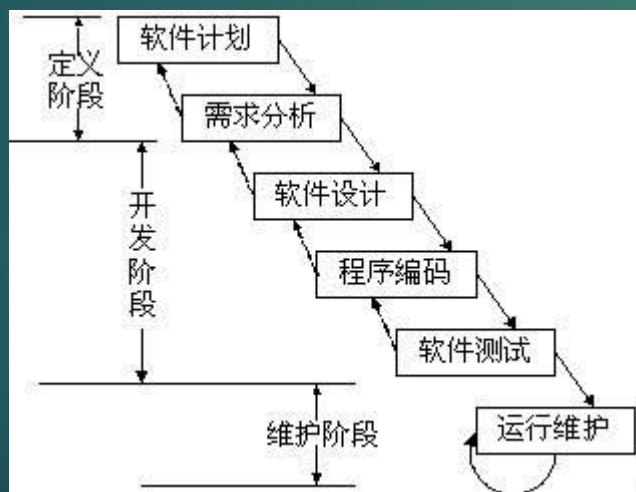
软件测试具体相关 测试设计来源



软件测试具体相关 开发模型

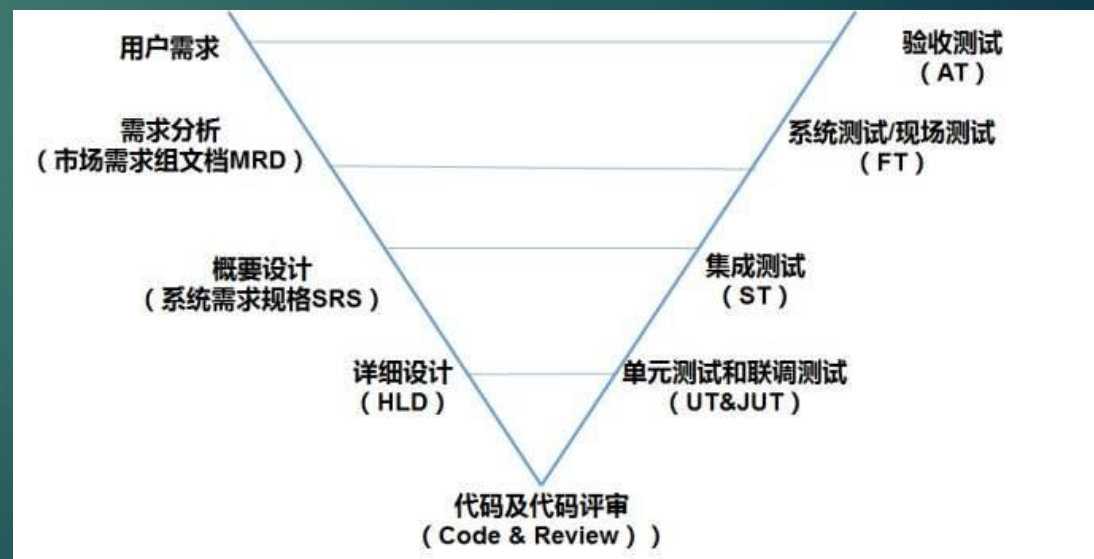
瀑布模型

瀑布模型是一个特别经典，甚至有点老套的周期模型，一般情况下将其分为计划、需求分析、概要设计、详细设计、编码以及单元测试、测试、运行维护等几个阶段。瀑布模型的周期是环环相扣的。每个周期中交互点都是一个里程碑，上一个周期的结束需要输出本次活动的工作结果，本次的活动的工作结果将会作为下一个周期的输入。



V模型

V模型从整体上看起来，就是一个V字型的结构，由左右两边组成。左边的下划线分别代表了需求分析、概要设计、详细设计、编码。右边的上划线代表了单元测试、集成测试、系统测试与验收测试。看起来V模型就是一个对称的结构，它的重要意义在于，非常明确的表明了测试过程中存在的不同的级别，并且非常清晰的描述了这些测试阶段和开发阶段的对应关系。



软件测试具体相关 理想模型



软件测试 总结

1. 白盒静态测试： PC-LINT9.0+ Source Insight 3.5
2. 人工走查： 软件设计工程师
3. 单元加覆盖： CUnit +Lcov 软件设计工程师
4. 黑盒测试： 测试工程师

尾声

保持创新，技术领先