第3章　C语言动态自动化单元测试框架

C语言单元测试工具主要包括PC-Lint，微软的VcTester、Visual Unit，Parasoft的C++Test、CUnit、Google Test等。本章详细介绍CUnit。

3.1　在Windows下安装C语言运行环境

Windows环境下不带C语言运行环境，如果需要在Windows下运行C语言，则要安装C语言的运行环境，这里介绍MinGW（Minimalist GNU for Windows）和MSYS2（Minimal SYStem 2）的安装。

3.1.1　安装配置MinGW

MinGW是一个可自由使用和自由发布的Windows特定头文件和使用GNU工具集导入的库的集合，它允许在GNU/Linux和Windows平台生成本地的Windows程序而不需要第三方C运行时（C Runtime）库。MinGW是一组包含文件和端口库的工具。MinGW的安装步骤如下。

1）安装mingw-w64-install.exe，本书安装的版本是8.1.0。

2）如果Windows是32位的，选择i686；如果Windows是64位的，选择x86_64。

3）配置环境变量MINGW_HOME。本书中的%MINGW_HOME%如下：

```
C:\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0\mingw64\
```

在PATH中加入：

```
%MINGW_HOME%[1]\bin\
%MINGW_HOME%\include\
```

4）打开命令行，运行如下命令：

```
C:\Users\xiang>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
```

① 本书使用%SOFTWAEW_HOME %来表示软件的HOME目录，下同。

```
COLLECT LTO WRAPPER=C:/mingw-w64/x86 64-8.1.0-posix-seh-rt v6-rev0/mingw64/
bin/../libexec/gcc/x86_64-w64-mingw32/8.1.0/lto-wrapper.exe
Target:x86 64-w64-mingw32
…
Thread model:posix
gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)
```

当出现“gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)”时，表示安装成功。

3.1.2　安装配置 MSYS2

MSYS2 是 MSYS 的一个独立改写版本，主要用于 shell 命令行开发环境，可以在 Windows 下执行 Linux 命令。同时，它也是一种在 Cygwin 和 MinGW-w64 基础上产生的，为寻求更好的互操作性的 Windows 软件。安装步骤如下。

1）下载“msys+7za+wget+svn+git+mercurial+cvs-rev13.7z”。

2）解压下载的软件。

3）将解压后的文件复制到“%MINGW_HOME%”目录下。

3.1.3　安装配置 IDE

1. 安装 IDE

在 Linux 下可以使用 vi 和 gedit 编写 C 语言代码，在 Windows 下可以使用写字板编写 C 语言代码，但是 Windows 中有许多优秀的 C 语言 IDE，这里以 Visual Studio Code 和 LLVM 为例（对于 Windows 操作系统，下面默认 64 位）。

1）下载并且安装 Visual Studio Code：VSCodeUserSetup-x64-1.46.0.exe。

2）下载并且安装 LLVM：LLVM-9.0.0-win64.exe。如图 3-1 所示，注意选择将 LLVM 加入所有系统用户的系统路径中。

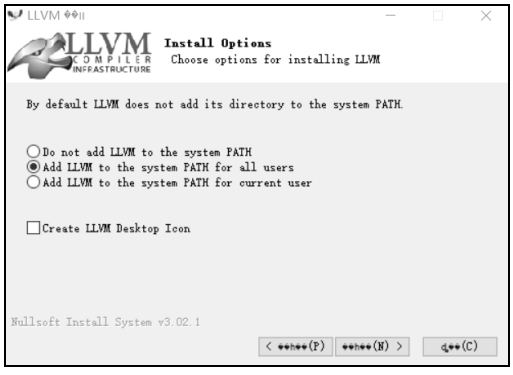

图 3-1　LLVM 安装界面

3）打开命令行，输入：

```
C:\MyC\process>clang
```

出现 clang:error:no input files，说明安装成功。

2. 安装插件

1）启动后安装中文插件。

在插件中找到“Chinese (Simplified) Language Pack for Visual Studio Code”并进行安装，如图 3-2 所示。

图 3-2　安装中文插件

2）安装支持 C/C++语言的插件。

在插件中找到“C/C++”并进行安装，如图 3-3 所示。

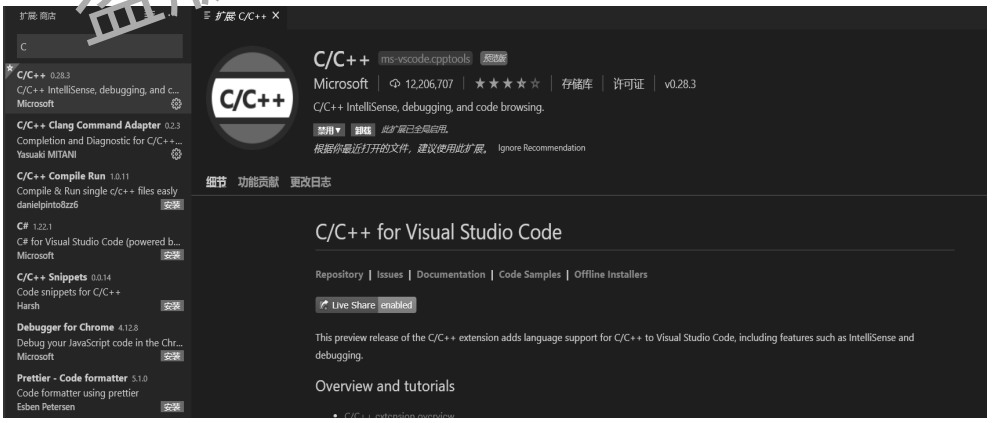

图 3-3　安装 C/C++语言插件

3）安装 Code Runner 插件。

在插件中找到“Code Runner”并进行安装，如图 3-4 所示。

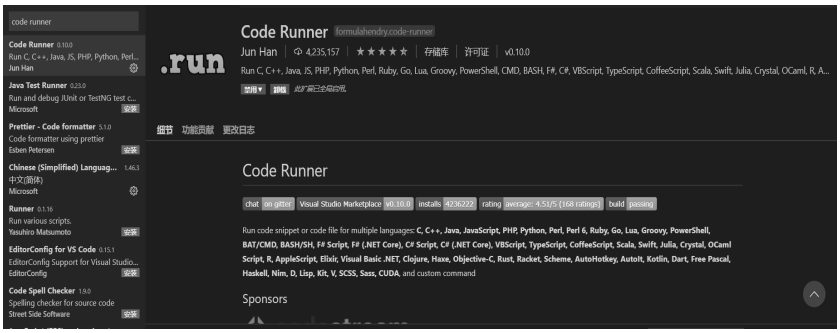

图 3-4　安装 Code Runner 插件

4）安装 C+/C++ Clang 命令插件。

在插件中找到“C+/C++ Clang Command Adapter”并进行安装，如图 3-5 所示。

图 3-5　安装 C+/C++ Clang 命令插件

5）在工作目录下创建.vscode 文件夹，并在文件夹中创建如下模板文件，如图 3-6 所示。

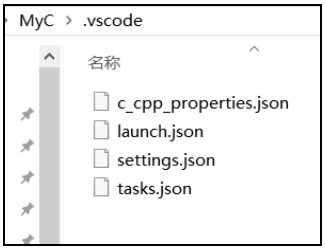

图 3-6　.vscode 文件夹

（1）c_cpp_properties.json：

```
{
    "configurations":[
        {
```

```
            "name":"MinGW",
            "intelliSenseMode":"gcc-x64",
            "compilerPath":"C:/mingw-w64/x86_64-8.1.0-posix-seh-rt_v6-rev0/
mingw64/bin/gcc.exe",
            "includePath":[
                "${workspaceFolder}"
            ],
            "defines":[],
            "browse":{
                "path":[
                    "${workspaceFolder}"
                ],
                "limitSymbolsToIncludedHeaders":true,
                "databaseFilename":""
            },
            "cStandard":"c11",
            "cppStandard":"c++17"
        }
    ],
    "version":4
}
```

其中，“C:/mingw-w64/x86_64-8.1.0-posix-seh-rt_v6-rev0/mingw64/”为%MinGW_HOME%目录。

（2）tasks.json：

```
// https://code.visualstudio.com/docs/editor/tasks
{
    "version":"2.0.0",
    "tasks":[
        {
            "label":"Compile",
            "command":"clang",
            "args":[
                "${file}",
                "-o",
                "${fileDirname}/${fileBasenameNoExtension}.exe",
                "-g",
                "-Wall",
                "-static-libgcc",
                "--target=x86_64-w64-mingw",
                "-std=c11" ,
            ],
            "type":"shell",
            "group":{
                "kind":"build",
                "isDefault":true
            },
            "presentation":{
                "echo":true,
                "reveal":"always",
```

```
            "focus":false,
            "panel":"shared"
        }
        // "problemMatcher":"$gcc"
    }
  ]
}
```

（3）launch.json：

```
// https://github.com/Microsoft/vscode-cpptools/blob/master/launch.md
{
  "version":"0.2.0",
  "configurations":[
    {
      "name":"(gdb)Launch",
      "type":"cppdbg",
      "request":"launch",
      "program":"${fileDirname}/${fileBasenameNoExtension}.exe",
      "args":[],
      "stopAtEntry":true,
      "cwd":"${workspaceFolder}",
      "environment":[],
      "externalConsole":true,
      "internalConsoleOptions":"neverOpen",
      "MIMode":"gdb",
      "miDebuggerPath":"gdb.exe",
      "setupCommands":[
        {
          "description":"Enable pretty-printing for gdb",
          "text":"-enable-pretty-printing",
          "ignoreFailures":false
        }
      ],
      "preLaunchTask":"Compile"
    }
  ]
}
```

（4）settings.json：

```
{
  "files.defaultLanguage":"c",
  "editor.formatOnType":true,
  "editor.snippetSuggestions":"top",

  "code-runner.runInTerminal":true,
  "code-runner.executorMap":{
    "c":"cd $dir && clang $fileName -o $fileNameWithoutExt.exe -Wall -g -Og
-static-libgcc -fcolor-diagnostics --target=x86 64-w64-mingw -std=c11 &&
$dir$fileNameWithoutExt",
    "cpp":"cd $dir && clang++ $fileName -o $fileNameWithoutExt.exe -Wall -g
-Og -static-libgcc -fcolor-diagnostics --target=x86_64-w64-mingw -std=c++17 &&
```

```
$dir$fileNameWithoutExt"
    },
    "code-runner.saveFileBeforeRun":true,
    "code-runner.preserveFocus":true,
    "code-runner.clearPreviousOutput":false,
    "C Cpp.clang format sortIncludes":true,
    "C Cpp.intelliSenseEngine":"Default",
    "C_Cpp.errorSquiggles":"Disabled",
    "C Cpp.autocomplete":"Disabled",

    "clang.cflags":[
        "--target=x86 64-w64-mingw",
        "-std=c11",
        "-Wall",
        "-lcunit"
    ],
    "clang.cxxflags":[
        "--target=x86 64-w64-mingw",
        "-std=c++17",
        "-Wall",
        "-lcunit"
    ],
    "clang.completion.enable":true,
    "files.associations":{
        "sstream":"c",
        "typeinfo":"c",
        "array":"c",
        "string":"c",
        "string_view":"c",
        "algorithm":"c"
    }
}
```


3. 验证

创建文件 HelloWorld.c：

```
#include <stdio.h>
int main(){
    printf("Hello World");
}
```

单击菜单“运行→以非调值模式运行”运行：

```
PS C:\Users\xiang> & 'c:\Users\xiang\.vscode\extensions\ms-vscode.
cpptools-1.13.7-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe'
'--stdin=Microsoft-MIEngine-In-0cous2n5.ty2' '--stdout=Microsoft-
MIEngine-Out-nusvphff.uwr' '--stderr=Microsoft-MIEngine-Error-iwpcqyvq.fic'
'--pid=Microsoft-MIEngine-Pid-blp34zz1.05e'
'--dbgExe=C:\mingw-w64\x86_64-8.1.0-posix-seh-rt_v6-rev0\mingw64\bin\gdb.ex
e' '--interpreter=mi'
Hello World
PS C:\Users\xiang>
```

3.2　安装编译 CUnit

3.2.1　在 Windows 下安装 CUnit

本书中的 Windows 命令均在 Windows 10 21H2 下验证通过。

1）下载“CUnit-2.1-3.rar”并解压到 C:\CUnit-2.1-3 目录下。

2）在%MinGW_HOME%\msys 下打开 msys.bat。

3）在 msys.bat 中依次运行如下命令：

```
cd C:\CUnit-2.1-3
$libtoolize
$automake --add-missing
$autoreconf
$./configure --prefix=/mingw
$make
$make install
```

4）复制%Mingw64_HOME%\msys\mingw\doc\CUnit\headers*.h 到%Mingw64_HOME%\include\目录下。

5）复制%Mingw64_HOME%\msys\mingw\lib\libcunit.a 到%Mingw64_HOME%\mingw64\lib\目录下。

6）进入工作目录（C:\MyC\process 为当前工作目录），运行如下命令：

```
C:\MyC\process>copy C:\CUnit-2.1-3\Share\*.dtd .\
C:\MyC\process>copy C:\CUnit-2.1-3\Share\*.xsl .\
```

3.2.2　在 Linux 下安装 CUnit

本书中的 Linux 命令均在 Ubuntu 18.04.6 下验证通过。

1）下载“CUnit-2.1-3.tar.bz2”并运行如下命令：

```
root@ubuntu:/#tar jxvf CUnit-2.1-3.tar.bz2
root@ubuntu:/#cd CUnit-2.1-3
root@ubuntu:/#aclocal //出现警告，不用理会
root@ubuntu:/#autoheader
root@ubuntu:/#autoconf
root@ubuntu:/#automake
root@ubuntu:/#automake --add-missing
root@ubuntu:/#libtoolize --automake --copy --debug --force
root@ubuntu:/#ls//查看是否生成 configure 文件。如果生成，则执行
root@ubuntu:/#./configure
root@ubuntu:/#ls//查看是否生成 makefile 文件。如果生成，则执行
root@ubuntu:/#automake --add-missing
root@ubuntu:/#make
root@ubuntu:/#sudo make install
```