

# VST 在 Win 10 下安装（利用 Cygwin）

## 准备工作

先下载 VST 的安装包：<https://github.com/PrincetonUniversity/VST>（或者直接 git clone），解压到某个目录备用。

然后下载对应版本（64 位或 32 位）的 Cygwin：<http://www.cygwin.com/>

安装 Cygwin，进入包安装界面（如下图），尝试安装下面几个包：`make`，`git`，`vim`。

The image shows the Cygwin package selection window. The top section is titled "Select Packages" with the subtitle "Select packages to install". Below this, there's a "View" dropdown set to "Pending", a search bar containing "make", and a "Clear" button. The main area is a table with columns: Package, Current, New, Bin?, Src?, and Categories. The table is currently empty, displaying "Nothing to install or update." Below this, there's another section with "View" set to "Full". The search bar still contains "make". The table below lists various packages related to make. A red circle highlights the dropdown arrow next to the "make" package, with a red text label "下拉选择版本安装" (Click to select version for installation) pointing to it.

Package	Current	New	Bin?	Src?	Categories
Nothing to install or update.					

  

Package	Current	New	Bin?	Src?	Categories
automake1.9	1.9.6-11	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
cmake	3.14.5-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
cmake-debuginfo	3.14.5-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Debug
cmake-doc	3.14.5-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
cmake-gui	3.14.5-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
emacs-cmake		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Editors
extra-cmake-modules	5.43.0-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
gcc-tools-epoch1-automake	1.9.6-2	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
gcc-tools-epoch2-automake	1.11.6-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
gcmakedep	1.0.3-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
imake		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Devel
imake-debuginfo		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Debug
libWMaker-devel		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Libs
libWMaker1		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Libs
libpagemaker-tools		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Graphics
libpagemaker0.0-debuginfo		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Debug
libpagemaker0.0-devel		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Libs
libpagemaker0.0-doc		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Libs
libpagemaker0.0_0		Skip	<input type="checkbox"/>	<input type="checkbox"/>	Libs
make	4.2.1-2	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel
make-debuginfo	4.2.1-2	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Debug
makedepend	1.0.6-1	Keep	<input type="checkbox"/>	<input type="checkbox"/>	Devel

勾选完成这些包之后点 Next 开始安装包，务必保持网络通畅。

## 相关环境设置

找到 Coq 安装目录下的 bin 文件夹，保证里面有 `coqc.exe`，记录该路径（如 `D:\Coq\bin`）。

然后在 VST 的根目录下新建文件 `CONFIGURE`，在里面写入下面的内容：

```
1 | COQBIN=/cygdrive/**你的 Coq bin 路径**
```

比如我的是 `D:\Coq\bin`，那么我就要向 `CONFIGURE` 中写入 `COQBIN=/cygdrive/D/Coq/bin/`。写完最好换行。

## 编译

打开 Cygwin 的终端。



定位到 VST 的根目录下。注意：要先执行 `cd /cygdrive` 进入 Cygwin 的虚拟目录，然后才能像在 CMD 中一样进行定位操作。这里以定位到 `D:\VST\` 为例。

```
asus@LAPTOP-S9UMPSPL ~
$ cd /cygdrive

asus@LAPTOP-S9UMPSPL /cygdrive
$ cd D

asus@LAPTOP-S9UMPSPL /cygdrive/D
$ cd VST

asus@LAPTOP-S9UMPSPL /cygdrive/D/VST
$
```

然后执行 `make floyd`，开始编译。（如果你的 CPU 有多个核心，可以执行 `make floyd -j3` 以加快速度）

编译完成后再执行 `make`，编译其他文件。

## 对相关库的支持

编译完成后，可以查看编译后的效果。

这里用 VST 的 Tutorial 和其使用到的 C 程序做示例。将这些文件放在 `D:\try` 中。

此电脑 > DATA (D:) > try

名称	修改日期	类型	大小
_CoqProject	2019/11/6 16:25	文件	1 KB
abs.c	2019/5/9 13:19	C 源文件	1 KB
abs.v	2019/5/9 13:58	Coq Script File	18 KB
add.c	2019/5/8 14:56	C 源文件	1 KB
add.v	2019/5/9 11:26	Coq Script File	18 KB
append.c	2019/5/20 15:56	C 源文件	1 KB
append.v	2019/5/20 15:56	Coq Script File	22 KB
fact.c	2019/5/10 13:49	C 源文件	1 KB
fact.v	2019/5/10 13:49	Coq Script File	19 KB
gcd.c	2019/5/10 15:23	C 源文件	1 KB
gcd.v	2019/5/10 15:25	Coq Script File	18 KB
max3.c	2019/5/8 17:03	C 源文件	1 KB
max3.v	2019/5/9 11:26	Coq Script File	18 KB
reverse.c	2019/5/20 15:56	C 源文件	1 KB
reverse.v	2019/5/20 15:56	Coq Script File	19 KB
swap.c	2019/5/8 17:19	C 源文件	1 KB
swap.v	2019/5/9 11:26	Coq Script File	20 KB
tri.c	2019/5/10 13:49	C 源文件	1 KB
tri.v	2019/5/10 13:49	Coq Script File	19 KB
Verified_Software_Toolchain.v	2019/11/14 15:49	Coq Script File	21 KB

然后在该目录下新建 `_CoqProject` 文件。里面输入以下内容：

```
1 -Q . PL
2 -Q ../VST/msl VST.msl
3 -Q ../VST/sepcomp VST.sepcomp
4 -Q ../VST/veric VST.veric
5 -Q ../VST/floyd VST.floyd
6 -Q ../VST/progs VST.progs
7 -R ../VST/compcert compcert
```

后面和 VST 有关的路径需要设定为 VST 的目录。这里使用的是 `D:\try`，且 VST 的根目录为 `D:\VST` 故这么写。

然后就可以在 Coq 中尝试编译各个 `.v` 文件，先编译各个 C 文件对应的 `.v` 文件，最后编译 VST 的 Tutorial 文件。

## 如果我想自己写 C 验证着玩呢？

可以考虑手动安装 Compcert 然后用。但是不推荐，如果只想要用 Compcert 的将 C 程序转换为 `.v` 文件的功能的话只要有编译好的 `clightgen.exe` 就行了。

要将 C 程序转换为 `.v` 文件，只需要将 C 程序和 `clightgen.exe` 和 `compcert.ini` 放在同一目录下，然后用控制台执行 `clightgen -normalize xxx.c` (`xxx.c` 是你的 C 程序名) 即可。

(不过并不清楚将一台机器上编译的 `clightgen` 放到另一台机器上有没有问题，可能是没什么问题的)

## 怎么证明？

从 Tutorial 就可以看出证明大概分三个部分：

1. 准备好需要验证的 C 程序，例如 `F.c`。一般我们希望证明的都是某个 C 函数的正确性，最后将这些对函数体的证明串起来，变成对 C 程序本身正确性的证明。

2. 用 clightgen 将其形式化，生成文件 `F.v`，其中包括了用 Coq 语言描述的 C 程序的 AST。将得到的 `F.v` 编译好，得到 `F.vo` 文件。
3. 写一个证明程序，记为 `verif_F.v`。`verif_F.v` 当中应该包含这些内容：
  1. 被证明程序的函数式实现。一般就是在 Coq 中用一系列语言表达被证明程序中的函数。这一部分往往还包括一些相关定理的证明。例如，我们要证明一个反转链表的函数的正确性，那么我们就可以在 Coq 中先定义列表，然后证明列表反转的一些性质，且这些性质最好是和我们程序中用到的算法相关的。
  2. 被证明程序的函数规格 (Function Specification)。简单来说，这一部分就是将 Coq 中的证明和 C 中的程序连接起来的桥梁。
  3. 对函数体的证明。这一部分在 Coq 中依靠 VST 实现。

当然，这是针对比较简单的情况。复杂如多个 C 程序的情况下也基本上分成这三个模块来做。

具体该怎么做可以参考 [Verifiable C](#) 和 VST/progs 下的各种 `verif_....v`。