

Integrating VectorCAST/Cover into a Make Environment

Introduction

With a valid working Makefile already established, it is possible and very beneficial to incorporate VectorCAST/Cover into the `Make` environment, to instrument for code coverage. An existing Makefile contains the knowledge required to create an executable program from your source files and libraries. It is most efficient, therefore, to use this same make file, to perform the VectorCAST/Cover code instrumentation.

The integration of VectorCAST/Cover into your `Make` environment will allow for a *'make instrumented'* command to automatically determine which source files in your program have changed, and invoke the VectorCAST command line interface (`clicast`) to instrument for code coverage. When the instrumented source files are recompiled and linked, the resultant executable program will generate the coverage data required for VectorCAST/Cover to provide post-execution analysis reports of the coverage achieved. This process is generic and can be performed with any compiler, platform, or version of `make`. The examples in this whitepaper were created using the GNU *'make'*.

Integrating VectorCAST/Cover into your `Make` environment involves creating a cover environment, selecting an instrumentation option, adding program source files to the cover environment, instrumenting the files, and finally compiling and linking files to create the executable program. The following sections provide a detailed explanation of each step.

Creating a New Cover Environment

The first step is to create a new cover environment. This environment contains dependencies that are added at instrumentation time and are subsequently needed for compiling and linking the executable. To create the environment the following VectorCAST/Cover command should be used:

```
$(VECTORCAST_DIR)/clicast Cover Environment Create <environment-name>
```

The `$(VECTORCAST_DIR)` variable in the command refers to the `VECTORCAST_DIR` environment variable that points to the VectorCAST installation directory.

Coverage Options

The next step after creating the cover environment is to select any options to be used for the project. When implementing code coverage in a `Make` environment, the more you must select the option to instrument files *"In_place"*. The VectorCAST/Cover command to instrument *In_place* is:

```
$(VECTORCAST_DIR)/clicast -e <env> Cover Options In_place Y
```

In_place instrumentation tells VectorCAST/Cover to create a backup of the original source file, and to store the instrumented version of the source file, into the original file name. As a result, existing references to the source file names in the Make file can remain unchanged, and can be used to build the instrumented executable program.

Other VectorCAST/Cover options can be set using similar clicast syntax. To see a list of all available options, use the command:

```
$(VECTORCAST_DIR)/clicast Cover Options
```

Adding Source Files to the Cover Environment

The VectorCAST/Cover command to add source files to an environment is:

```
$(VECTORCAST_DIR)/clicast -e <env> Cover Source Add <file.extension>
```

As of the release of VectorCAST 4.1d, the command to add source files can be modified to add all or an arbitrary number of source files in a working directory to a cover environment. The wildcard notation: *.extension can be used in place of <file.extension> to add all source files in the directory. Multiple unit names can also be listed in the command to add a select number of source files. For example, to add all C++ source files in a particular directory use:

```
$(VECTORCAST_DIR)/clicast -e <env> Cover Source Add *.cpp
```

To add two source files at the same time use:

```
$(VECTORCAST_DIR)/clicast -e <env> Cover Source Add one.cpp two.cpp
```

Note: *In_place* instrumentation will create backup (.bak) files of the original source files. the VectorCAST/Cover “uninstrument” command (discussed in the next section) will restore the file to its original (uninstrumented) state.

Instrumenting a Source File

The final step before compiling and linking the executable is to select the type of coverage. The following are the five command options to instrument for the various types of code coverage:

```
$(VECTORCAST_DIR)/clicast -e <env> [-u <unit>] Cover Instrument State
```

```
$(VECTORCAST_DIR)/clicast -e <env> [-u <unit>] Cover Instrument Branch
```

```
$(VECTORCAST_DIR)/clicast -e <env> [-u <unit>] Cover Instrument Mcdc
```

```
$(VECTORCAST_DIR)/clicast -e <env> [-u <unit>] Cover Instrument LEVELB
```

```
$(VECTORCAST_DIR)/clicast -e <env> [-u <unit>] Cover Instrument LEVELA
```

The command to un-instrument coverage is:

```
$(VECTORCAST_DIR)/clicast -e <env> [-u <unit>] Cover Instrument NONE
```

The [-u <unit>] is optional with each command. If [-u <unit>] is used, the command applies to that source file only, if [-u <unit>] is not used the command applies to all files in the VectorCAST/Cover environment.

Compile and Link

After adding the source files to the cover environment and performing coverage instrumentation, the source files need to be compiled and linked to create the instrumented executable program. Along with those source files will also be the compilation and linking of the appropriate `c_cover_io.cpp` (or `c_cover_io.c`) file, which is located in the cover environment directory.

The command to do this, will be the same make command used to build the application without coverage instrumentation (e.g. “make -f Makefile”)

Sample Makefile

The following sample Makefile, developed using the GNU ‘make’, uses the VectorCAST tutorial source code to demonstrate how VectorCAST/Cover can be integrated into a make environment. In the example, ‘make’ simply creates a tutorial executable after compiling and linking the source and object files. When ‘make *instrumented*’ is performed, the source files are added to a cover environment, instrumented for statement coverage, compiled and linked to create an instrumented executable program.

```
##### EXISTING MAKEFILE
#####

#--- Existing Makefile in this example doesn't contain any Includes

INCLUDES =

source_files = manager.cpp database.cpp manager_driver.cpp
obj_files = $(patsubst %.cpp, %.o, $(source_files))

tutorial.exe : $(obj_files) $(source_files)
    g++ -o tutorial.exe $(obj_files)

%.o : %.cpp
g++ -c $< $(INCLUDES) -o $@

all : tutorial.exe

clean :
    rm -f tutorial.exe $(obj_files)
```

```

##### INTEGRATING VectorCAST/COVER
#####

#--- Set the Cover Environment variable name.

VCAST_COVER_ENV = CoverENV

#--- The Instrumented target below adds the Cover environment
#--- directory as an Include. The object file dependency needed for
#--- linking the instrumented executable is then added to the existing
#--- set of object files. Lastly, the instrumented target lists each
#--- prerequisite target needed for the integration process.

instrumented : INCLUDES += -I $(VCAST_COVER_ENV)
instrumented : obj_files += c_cover_io.o
instrumented : $(VCAST_COVER_ENV) vcast_set_option add_source inst \
              c_cover_io.o all

#--- The c_cover_io.o target below compiles the dependency source file
#--- in the cover environment directory to create the object file
#--- needed for linking the instrumented executable.

c_cover_io.o : $(VCAST_COVER_ENV)/c_cover_io.cpp
              g++ -c $< $(INCLUDES) -o $@

#----- Create New Cover Environment -----#

$(VCAST_COVER_ENV) : $(VECTORCAST_DIR)
                    $(VECTORCAST_DIR)/clicast Cover Environment Create
$(VCAST_COVER_ENV)

#----- Coverage Option -----#

vcast_set_option : $(VECTORCAST_DIR) $(VCAST_COVER_ENV)
                  $(VECTORCAST_DIR)/clicast -e $(VCAST_COVER_ENV) Cover Options
In_place Y

#----- Add Source Files to Cover Environment -----#

add_source : $(VECTORCAST_DIR) $(VCAST_COVER_ENV)
             $(VECTORCAST_DIR)/clicast -e $(VCAST_COVER_ENV) Cover Source Add
$(source_files)

#----- Instrument Coverage -----#

inst: $(VECTORCAST_DIR) $(VCAST_COVER_ENV)
      $(VECTORCAST_DIR)/clicast -e $(VCAST_COVER_ENV) Cover Instrument
Statement

```

```
##### UN-INSTRUMENT COVERAGE
#####
```

```
uninstrumented : obj_files += c_cover_io.o
```

```
uninstrumented :
    $(VECTORCAST_DIR)/clicast -e $(VCAST_COVER_ENV) Cover Instrument
None
    rm -f tutorial.exe $(obj_files)
```

```
#----- Phony Target -----#
```

```
.PHONY : clean instrumented uninstrumented vcast_set_option inst \
add_source
```