Getting Started with CUT

Sergei Gnezdov

November 11, 2004

1 Introduction

Cut is a unit testing framework for C, C++ and Objective-C. This document covers version 2.3 of cut. Our goal is to help you to get started with cut.
To download cut go to http://sourceforge.net/projects/cut/

2 Getting Ready

Cut comes in source code form. You will have to compile it to start using. You should have no problem to compile it on BSD (FreeBSD) or Linux machine.
To compile run

```
make
```

command without arguments. This should create `cutgen`application.

3 Test Driven Development in C with cut

3.1 Overview

To show the usage of the cut we will create a very simple C project. You will have to create a directory for your project. It is referenced as Project directory further on. The directory will contain the following files:

- Makefile
- Main.c
- Compute.h

- TestCompute.c
- Compute.c
- cut.h

3.2 Creating Makefile

Create Makefile with the following content:

```
CC = gcc
LD = gcc
MODULES = Compute
OBJS = $(MODULES:%=%.o)
TESTS = $(MODULES:%=Test%.c)
TESTS_OBJS = $(MODULES:%=Test%.o)
LIBS =
CCOPTS = -c
DEBUG =

# default target
help:
	echo "Type 'make application' to build the application."
	echo "Type 'make check' to biuld the application."

# top level target to create production application
application: $(OBJS) Main.o
	$(LD) $(LDOPTS) $(MODULES:%=%.o) Main.o $(LIBS) -o app

# top level target to run test cases
test: cutcheck
	./cutcheck

# top level target to clean up the project directory
clean:
	-rm *.o app cutcheck

cutcheck.c: $(TESTS)
	cutgen -o cutcheck.c $(TESTS)

cutcheck: $(OBJS) $(TESTS_OBJS) cutcheck.o
	$(LD) $(LDOPTS) $(OBJS) $(TESTS_OBJS) $(LIBS)\
```

```
        cutcheck.o -o cutcheck

.c.o:
        $(CC) $(CCOPTS) $(DEBUG) -o $@ $<
```

3.3 Defining Sum Function

Compute.h header defines a function we want to test:

```
#ifndef COMPUTE_H_INCLUDED
#define COMPUTE_H_INCLUDED

/* Summarizes two numbers */
int sum(int, int);

#endif
```

3.4 Creating Test Case

For consitency the test file name is based on the name of the header file beeing tested. Append Test prefix and replace h with c extension. TestCompute.c file has the following content:

```
#include <stdio.h>
#include "cut.h"
#include "Compute.h"

void __CUT__Sum( void )
{
  ASSERT(3 == sum(1,2), "Check sum");
}
```

TestCompute.c does not compile yet. See it for yourself with the following command:

```
make TestCompute.o
```

The compileation fails because the compiler can't find cut.h file. Copy cut.h file from the cut-2.3 project directory into our project directory.
Run

```
make TestCompute.o
```

again. This should create TestCompute.o file.

4 Implementing sum Function

Create Compute.c file:

```
int sum(int a, int b)
{
  return 0;
}
```

Run

```
make test
```

now. It should fail with the message similar to the following:

```
TestCompute.c(7): Check sum
TestCompute.c(7): Failed expression: 3 == sum(1,2)
```

This is because we did not implement sum function yet.
Change Compute.c to contain the following code:

```
int sum(int a, int b)
{
  return a+b;
}
```

and run `make test` again. The test should pass now.

4.1 Creating main application

Since our function operates as expected we can create a main application now:

```
#include <stdio.h>
#include "Compute.h"

int main(void)
{
  printf("%i\n", sum(3,2));
  return 0;
}
```

Run `make application` to see the result.