**Cmocka**

Unit testing framework

Cmocka is an elegant unit testing framework for C with support for mock objects. It only requires the standard C library, works on a range of computing platforms (including embedded) and with different compilers.

Official Website: https://cmocka.org

**How to install:**

This library is available for both Windows and Linux.

For LINUX:

* First download [cmocka-1.1.0.tar.xz](https://cmocka.org/files/1.1/cmocka-1.1.0.tar.xz)
* Install it using the following commands

$ cd ~/Downloads

$ xz –d cmocka-1.1.0.tar.xz

$ tar vxf cmocka-1.1.0.tar

$ cd cmocka-1.1.0

$ mkdir build

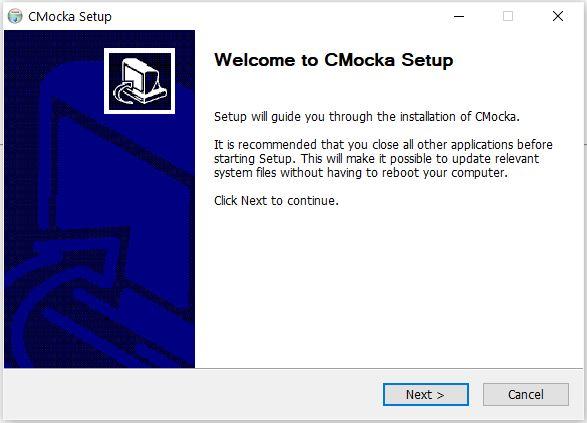
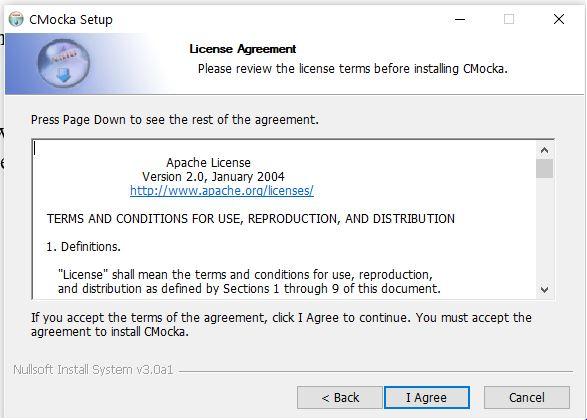
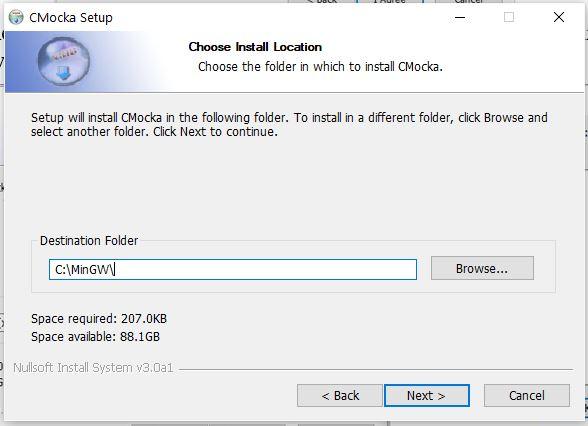
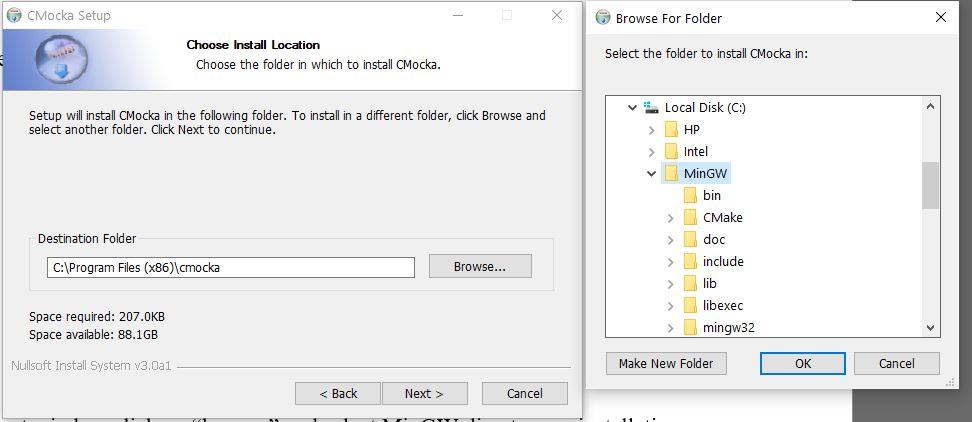
$ cd build

$ cmake -DCMAKE\_INSTALL\_PREFIX=/usr -DCMAKE\_BUILD\_TYPE=Release ..

$ make

$ sudo make install

For Windows

* First download [cmocka-1.1.0-mingw.exe](https://cmocka.org/files/1.1/cmocka-1.1.0-mingw.exe)
* Run the executable you will get the following window
* Click “next” and click on “I Agree”.
* On the next window click on “browse” and select MinGW directory as installation directory.
* Click “next” and click “install” and you’re done.

**How to write Test program:**

Using cmocka is very easy, follow the below example.

Str\_length.c

#include <assert.h>

#include <stdlib.h>

extern void mock\_assert(const int result, const char\* const expression,

const char \* const file, const int line);

#undef assert

#define assert(expression) \

mock\_assert(((expression) ? 1 : 0), #expression, \_\_FILE\_\_, \_\_LINE\_\_);

int str\_length(char \*str)

{

assert(str != NULL);

int count = 0;

while(\*str != '\0') {

count += 1;

str++;

}

if(count > 0)

return count;

else

return -1;

}

Suppose we want to test the above code with cmocka. In the source code assert functions should be replaced with cmocka mock\_assert function, as it will abort the program. Test code for str\_length.c is given below.

str\_length\_test.c

#include <stdlib.h>

#include <string.h>

#include <stdarg.h>

#include <stddef.h>

#include <setjmp.h>

#include <cmocka.h>

extern int str\_length(char \*str);

static char s[10];

void init()

{

s[0] = 'a';

s[1] = 'b';

s[2] = 'c';

s[3] = 'd';

}

void null\_test(void \*\*state)

{

expect\_assert\_failure(str\_length(NULL));

}

void unterminated\_test(void \*\*state)

{

assert\_int\_equal(str\_length(s), 4);

}

void terminated\_test(void \*\*state)

{

s[4] = '\0';

assert\_int\_equal(str\_length(s), 4);

}

int main(int argc, char \*argv[])

{

init();

const struct CMUnitTest tests[] = {

cmocka\_unit\_test(null\_test),

cmocka\_unit\_test(unterminated\_test),

cmocka\_unit\_test(terminated\_test),

};

return cmocka\_run\_group\_tests(tests, NULL, NULL);

}

**Test Cases:**

Math Functions:

|  |  |  |
| --- | --- | --- |
| **Function file** | **Test function file** | **How to build** |
| evenOrOdd.c | evenOrOdd\_test.c | gcc evenOrOdd.c enenOrOdd\_test.c -lcmocka |
| factorial.c | Factorial\_test.c | gcc factorial.c factorial\_test.c -lcmocka |
| factors.c | factors\_test.c | gcc factors.c factors\_test.c -lcmocka |
| fibonacci.c | fibonacci\_test.c | gcc fibonacci.c fibonacci\_test.c -lcmocka |
| FV.c | FV\_test.c | gcc FV.c FV\_test.c -lcmocka |
| gcd.c | gcd\_test.c | gcc gcd.c gcd\_test.c -lcmocka |
| is\_prime.c | is\_prime\_test.c | gcc is\_prime.c is\_prime\_test.c -lcmocka |
| lcm.c | lcm\_test.c | gcc lcm.c lcm\_test.c -lcmocka |
| nCr.c | nCr\_test.c | gcc nCr.c nCr\_test.c -lcmocka |
| nPr.c | nPr\_test.c | gcc nPr.c nPr\_test.c -lcmocka |
| power.c | power\_test.c | gcc power.c power\_test.c -lcmocka |
| primefactor.c | primefactor\_test.c | gcc primefactor.c primefactor\_test.c -lcmocka |
| PV.c | PV\_test.c | gcc PV.c PV\_test.c -lcmocka |

String Functions

|  |  |  |
| --- | --- | --- |
| **Function File** | **Test function file** | **How to build** |
| mem\_copy.c | mem\_copy\_test.c | gcc mem\_copy.c mem\_copy\_test.c -lcmocka |
| str\_compare.c | str\_compare\_test.c | gcc str\_compare.c str\_compare\_test.c -lcmocka |
| str\_copy.c | str\_copy\_test.c | gcc str\_copy.c str\_copy\_test.c -lcmocka |
| str\_cpy.c | str\_cpy\_test.c | gcc str\_cpy.c str\_cpy\_test.c -lcmocka |
| str\_find\_char.c | str\_find\_char\_test.c | gcc str\_find\_char.c str\_find\_char\_test.c -lcmocka |
| str\_length.c | str\_length\_test.c | gcc str\_length.c str\_length\_test.c -lcmocka |
| str\_split.c | str\_split\_test.c | gcc str\_split.c str\_split\_test.c -lcmocka |
| str\_find\_substring.c | str\_find\_substring\_test.c | gcc str\_find\_substring.c str\_find\_substring\_test.c -lcmocka |

Generic Sort Function

|  |  |  |
| --- | --- | --- |
| **Function File** | **Test function file** | **How to build** |
| gensort.c | gensort\_test.c | gcc gensort.c gensort\_test.c mem\_copy.c -lcmocka |

Linked-list Functions

|  |  |  |
| --- | --- | --- |
| **Function File** | **Test function file** | **How to build** |
| AlternatingSplit.c | alternatingSplit\_test.c | gcc AlternatingSplit.c alternatingSplit\_test.c MoveNode.c -lcmocka |
| Append.c | Append\_test.c | gcc Append.c Append\_test.c -lcmocka |
| count.c | count\_test.c | gcc count.c count\_test.c -lcmocka |
| countNode.c | countNode\_test.c | gcc countNode.c countNode\_test.c -lcmocka |
| DeleteList.c | DeleteList\_test.c | gcc DeleteList.c DeleteList\_test.c -lcmocka |
| FrontBackSplit.c | FrontBackSplit\_test.c | gcc FrontBackSplit.c FrontBackSplit\_test.c countNode.c -lcmocka |
| GetNth.c | GetNth\_test.c | gcc GetNth.c GetNth\_test.c -lcmocka |
| InsertNth.c | InsertNth\_test.c | gcc InsertNth.c InsertNth\_test.c -lcmocka |
| InsertSort.c | InsertSort\_test. | gcc InsertSort.c InsertSort\_test.c SortedInsert.c -lcmocka |
| MergeSort.c | MergSort\_test.c | gcc MergSort\_test.c MergeSort.c FrontBackSplit.c SortedMerge.c MoveNode.c countNode.c -lcmocka |
| MoveNode.c | MoveNode\_test.c | gcc MoveNode.c MoveNode\_test.c -lcmocka |
| newNode.c | newNode\_test.c | gcc newNode.c newNode\_test.c -lcmocka |
| pop.c | pop\_test.c | gcc pop.c pop\_test.c -lcmocka |
| push.c | push\_test.c | gcc push.c push\_test.c -lcmocka |
| RemoveDuplicates.c | RemoveDuplicates\_test.c | gcc RemoveDuplicates.c RemoveDuplicates\_test.c -lcmocka |
| Reverse.c | Reverse\_test.c | gcc Reverse.c Reverse\_test.c -lcmocka |
| ShuffleMerge.c | ShuffleMerge\_test.c | gcc ShuffleMerge.c ShuffleMerge\_test.c MoveNode.c -lcmocka |
| SortedInsert.c | SortedInsert\_test.c | gcc SortedInsert.c SortedInsert\_test.c -lcmocka |
| SortedIntersect.c | SortedIntersect\_test.c | gcc SortedIntersect.c SortedIntersect\_test.c push.c -lcmocka |
| SortedMerge.c | SortedMerge\_test.c | gcc SortedMerge.c SortedMerge\_test.c MoveNode.c -lcmocka |