

Hogyan tesztelünk?

Unit tesztünk alapja a main függvény visszatérése. Minden egyes unithoz írunk egy külön test c-fájlt, amelyben megvizsgáljuk a metódusait. Ha 0-val tér végül vissza, akkor a teszt sikeresen lefutott, ellenkező esetben hiba volt.

Előkészületek:

A faup project cmake-et használ. Feltérképezve a könyvtársturktúrát, az src mappán belül találjuk a lib könyvtárat, amiből tesztelni fogunk, és a a tests mappát, ahol a teszt fájlok, és a hozzá kapcsolódó konfigurációk láthatók. Számunkra fontos a CmakeLists.txt, ahol megadhatjuk a futtatandó teszteket.



```
CMakeLists.txt (~//faup-master/src/tests) - gedit

set(TEST_FAUP_SCRIPT "${faup-project_BINARY_DIR}/src/tests/test_faup.sh")
set(URLS_JSON "${faup-project_SOURCE_DIR}/src/tests/urls.json")

add_test(Url_Argument      ${TEST_FAUP_SCRIPT} Url_Argument)
add_test(File_Argument     ${TEST_FAUP_SCRIPT} File_Argument)

# Cover all the issues that we had (if applicable)
add_test(Issue24           ${TEST_FAUP_SCRIPT} issue 24)
add_test(Issue27           ${TEST_FAUP_SCRIPT} issue 27)
add_test(Issue28           ${TEST_FAUP_SCRIPT} issue 28)
add_test(Issue30           ${TEST_FAUP_SCRIPT} issue 30)
add_test(Issue69           ${TEST_FAUP_SCRIPT} issue 69)

# Some C tests
include_directories(AFTER ${FAUP_INCLUDE_DIRS})

if(LINUX)
    add_executable(multithreads multithreads.c)
    target_link_libraries(multithreads ${FAUP_LIBRARY} "pthread"
${CMAKE_DL_LIBS})
    add_test(MultiThreads ${TEST_FAUP_SCRIPT} multithreads)
endif(LINUX)

# C API/ABI issues
add_executable(issue36 issue36.c)
target_link_libraries(issue36 ${FAUP_LIBRARY} ${CMAKE_DL_LIBS})
add_test(Issue36 issue36)

# Snapshot Tests
add_executable(snapshot snapshot.c)
target_link_libraries(snapshot ${FAUP_LIBRARY} ${CMAKE_DL_LIBS})
add_test(Snapshot snapshot)

# Hash Table Tests
add_executable(hashtable hashtable.c)
target_link_libraries(hashtable ${FAUP_LIBRARY} ${CMAKE_DL_LIBS})
add_test(Hashtable hashtable)

# JSON Urls test
add_executable(json-tester json-tester.c)
target_link_libraries(json-tester ${FAUP_LIBRARY} ${CMAKE_DL_LIBS})
add_test(JsonURLsTester json-tester ${URLS_JSON})
```

A tesztek futtatásához a ctest toolt fogjuk használni, amely a cmake mellé jár alapértelmezésként. Első lépésként tudnunk kell, hogy hogyan fordítjuk le a projectet.

Fordítás:

A README-ből megtudtam, hogy az első fordításhoz ki kell törölni a cmake cache fájlt a fő mappából. Ezek után létrehozunk egy build mappát, és belépünk oda. Nyitok egy terminal-t, és a következő parancsokat adom ki benn:

- cmake
- make
- (sudo make install)

Ez utóbbi csak a telepítéshez szükséges. A cmake megcsinálja a megfelelő konfigurációt, majd a make lefordítja a faup-ot.

Tesztek futtatása:

Hogy lássuk hol állunk, az előző lépések folytatásaként adjuk ki a ctest parancsot a build könyvtárban

```
fox@Foxy: ~/faup-master/build
-- Checking for module 'caca'
-- No package 'caca' found
Libcaca was not found, faup will not offer snapshot browsing
CMake system: Linux
Discovered distribution type: debian
Version: 1.6
Version Major: 1
Version Minor: 6
Mandir: share/man
-- Configuring done
-- Generating done
-- Build files have been written to: /home/fox/faup-master/build
fox@Foxy:~/faup-master/build$ make
[ 39%] Built target faup_static
[ 78%] Built target faupl
[ 84%] Built target faup
[ 88%] Built target json-tester
[ 92%] Built target hashtable
[ 96%] Built target snapshot
[100%] Built target issue36
fox@Foxy:~/faup-master/build$ ctest
Test project /home/fox/faup-master/build
  Start 1: Url_Argument
1/11 Test #1: Url_Argument ..... Passed    0.11 sec
  Start 2: File_Argument
2/11 Test #2: File_Argument ..... Passed    0.11 sec
  Start 3: Issue24
3/11 Test #3: Issue24 ..... Passed    0.11 sec
  Start 4: Issue27
4/11 Test #4: Issue27 ..... Passed    0.11 sec
  Start 5: Issue28
5/11 Test #5: Issue28 ..... Passed    0.11 sec
  Start 6: Issue30
6/11 Test #6: Issue30 ..... Passed    0.11 sec
  Start 7: Issue69
7/11 Test #7: Issue69 ..... Passed    0.11 sec
  Start 8: Issue36
8/11 Test #8: Issue36 ..... Passed    0.01 sec
  Start 9: Snapshot
9/11 Test #9: Snapshot ..... Passed    0.01 sec
  Start 10: Hashtable
10/11 Test #10: Hashtable ..... Passed    0.00 sec
  Start 11: JsonURLsTester
11/11 Test #11: JsonURLsTester ..... Passed    0.01 sec

100% tests passed, 0 tests failed out of 11

Total Test time (real) = 0.78 sec
fox@Foxy:~/faup-master/build$
```

11 unit tesztje van készen, melyek közül mindegyik sikeresen lefutott.

Új tesztek írása:

A src mappából kiválasztottam a lib/decode.c – t. Ezt a unitot fogom letesztelni a bemutatóm keretében. Ehhez első lépésként beírom a tests mappa CMakeLists.txt fájlába a következőket:

```
# Decode test
add_executable(decode-tester decode-tester.c)
target_link_libraries(decode-tester ${FAUP_LIBRARY} ${CMAKE_DL_LIBS})
add_test(DecodeTester decode-tester)
```

Ezzel megmondom a cmake-nek, hogy adjon hozzá DecodeTester névvel egy új tesztet, amely forrása a decode-tester.c fájlban helyezkedik el.

Írjuk meg a decode.c-t, amely a tesztünket tartalmazza. Első lépésként csak visszatérek egy üres main függvényből 0-val. A bevezetés fényében ennek sikeres tesztnek kell lennie.

Elmentem, és belépek a fő mappában lévő build mappába. Kiadom a ctest parancsot, hogy lefuttassam a teszteket. Ekkor csak az eddig 11 tesztet látom. Kiadom a cmake, majd utána a ctest parancsot. A következő fogad:

```

fox@Foxy: ~/faup-master/build
1/12 Test #1: Url_Argument ..... Passed    0.14 sec
      Start 2: File_Argument
2/12 Test #2: File_Argument ..... Passed    0.10 sec
      Start 3: Issue24
3/12 Test #3: Issue24 ..... Passed    0.11 sec
      Start 4: Issue27
4/12 Test #4: Issue27 ..... Passed    0.11 sec
      Start 5: Issue28
5/12 Test #5: Issue28 ..... Passed    0.10 sec
      Start 6: Issue30
6/12 Test #6: Issue30 ..... Passed    0.12 sec
      Start 7: Issue69
7/12 Test #7: Issue69 ..... Passed    0.12 sec
      Start 8: Issue36
8/12 Test #8: Issue36 ..... Passed    0.01 sec
      Start 9: Snapshot
9/12 Test #9: Snapshot ..... Passed    0.00 sec
      Start 10: Hashtable
10/12 Test #10: Hashtable ..... Passed    0.00 sec
      Start 11: JsonURLsTester
11/12 Test #11: JsonURLsTester ..... Passed    0.01 sec
      Start 12: Mytest
Could not find executable mytest
Looked in the following places:
mytest
mytest
Release/mytest
Release/mytest
Debug/mytest
Debug/mytest
MinSizeRel/mytest
MinSizeRel/mytest
RelWithDebInfo/mytest
RelWithDebInfo/mytest
Deployment/mytest
Deployment/mytest
Development/mytest
Development/mytest
Unable to find executable: mytest
12/12 Test #12: Mytest .....***Not Run    0.00 sec

92% tests passed, 1 tests failed out of 12

Total Test time (real) =  0.84 sec

The following tests FAILED:
    12 - Mytest (Not Run)
Errors while running CTest
fox@Foxy:~/faup-master/build$

```

(megj: amikor fotóztam, akkor még Mytest-nek hívtam, utána pedig rájöttem mi a hiba a decode-dal, és már nem tudtam reprodukálni :))

Miért van ez a hiba?

Több óra kísérletezés után végül Áron hozta a megvilágosodást. Amikor elmagyaráztam neki a dolgot, akkor jött a megvilágosodás, és emiatt építettem így föl a leírásom. Amit nem számoltam bele, hogy a cmake önmagában nem fordítja le a kódot, így a helyes sorrend a következő:

- cmake ..
- make
- ctest

Ezután már a bővült tesztgyűjteményt látjuk, ráadásul le is fut a teszt:

```
fox@Foxy: ~/faup-master/build
-- Checking for module 'lua5.3>=5.1'
--
-- Checking for module 'lua5.2>=5.1'
--
-- Checking for module 'lua5.1>=5.1'
--
-- Checking for module 'caca'
-- No package 'caca' found
Libcaca was not found, faup will not offer snapshot browsing
CMake system: Linux
Discovered distribution type: debian
Version: 1.6
Version Major: 1
Version Minor: 6
Mandir: share/man
-- Configuring done
-- Generating done
-- Build files have been written to: /home/fox/faup-master/build
fox@Foxy:~/faup-master/build$ ctest
Test project /home/fox/faup-master/build
  Start 1: Url_Argument
1/12 Test #1: Url_Argument ..... Passed    0.12 sec
  Start 2: File_Argument
2/12 Test #2: File_Argument ..... Passed    0.10 sec
  Start 3: Issue24
3/12 Test #3: Issue24 ..... Passed    0.11 sec
  Start 4: Issue27
4/12 Test #4: Issue27 ..... Passed    0.10 sec
  Start 5: Issue28
5/12 Test #5: Issue28 ..... Passed    0.10 sec
  Start 6: Issue30
6/12 Test #6: Issue30 ..... Passed    0.10 sec
  Start 7: Issue69
7/12 Test #7: Issue69 ..... Passed    0.10 sec
  Start 8: Issue36
8/12 Test #8: Issue36 ..... Passed    0.01 sec
  Start 9: Snapshot
9/12 Test #9: Snapshot ..... Passed    0.00 sec
  Start 10: Hashtable
10/12 Test #10: Hashtable ..... Passed    0.00 sec
  Start 11: JsonURLsTester
11/12 Test #11: JsonURLsTester ..... Passed    0.01 sec
  Start 12: DecodeTester
12/12 Test #12: DecodeTester ..... Passed    0.00 sec

100% tests passed, 0 tests failed out of 12

Total Test time (real) =  0.76 sec
fox@Foxy:~/faup-master/build$
```

Győződjünk meg a helyességről. Átírom a decode.c-t, hogy 1-gyel térjen vissza a main függvénye. Majd lefordítom, és futtatom a tesztet a fent ismertetett módon.

```
fox@Foxy: ~/faup-master/build
[ 84%] Built target json-tester
[ 88%] Built target hashtable
[ 92%] Built target snapshot
[ 96%] Built target issue36
Scanning dependencies of target decode-tester
[ 98%] Building C object src/tests/CMakeFiles/decode-tester.dir/decode-tester.c.o
/home/fox/faup-master/src/tests/decode-tester.c: In function 'main':
/home/fox/faup-master/src/tests/decode-tester.c:12:8: warning: implicit declaration of function 'is_ipv4' [-Wimplicit-function-declaration]
    if (!is_ipv4("192.168.1.12", 12)){
        ^
[100%] Linking C executable decode-tester
[100%] Built target decode-tester
fox@Foxy:~/faup-master/build$ ctest
Test project /home/fox/faup-master/build
  Start 1: Url_Argument
1/12 Test #1: Url_Argument ..... Passed    0.13 sec
  Start 2: File_Argument
2/12 Test #2: File_Argument ..... Passed    0.10 sec
  Start 3: Issue24
3/12 Test #3: Issue24 ..... Passed    0.10 sec
  Start 4: Issue27
4/12 Test #4: Issue27 ..... Passed    0.10 sec
  Start 5: Issue28
5/12 Test #5: Issue28 ..... Passed    0.10 sec
  Start 6: Issue30
6/12 Test #6: Issue30 ..... Passed    0.10 sec
  Start 7: Issue69
7/12 Test #7: Issue69 ..... Passed    0.10 sec
  Start 8: Issue36
8/12 Test #8: Issue36 ..... Passed    0.01 sec
  Start 9: Snapshot
9/12 Test #9: Snapshot ..... Passed    0.00 sec
  Start 10: Hashtable
10/12 Test #10: Hashtable ..... Passed    0.00 sec
  Start 11: JsonURLsTester
11/12 Test #11: JsonURLsTester ..... Passed    0.01 sec
  Start 12: DecodeTester
12/12 Test #12: DecodeTester .....***Failed    0.00 sec

92% tests passed, 1 tests failed out of 12

Total Test time (real) =  0.76 sec

The following tests FAILED:
   12 - DecodeTester (Failed)
Errors while running CTest
fox@Foxy:~/faup-master/build$
```

Értékelés: ezt az eredményt vártuk. Meggyőződhattünk róla, hogy tényleg a main visszatéréstől függ a teszt passed vagy failed értékelése.

Írjunk valós tesztet

Ehhez include-olom a faup/decode headerjét, és meghívom a tesztelni való függvényeit. A példában az is_ipv4-et választottam ennek bemutatására, mely bemenetként egy karakterláncot, és ennek hosszát várja, és megállapítja, hogy a megadott lánc ipv4 cím-e.

A tesztelési stratégiám a következő:

- Megnézem egy korrekt ip – korrekt hossz párosra. (várt: true)
- Megnézem egy korrekt ip – (másik) korrekt hossz párosra. (várt: true)
- Megnézem egy korrekt ip – túl hosszú hossz párosra. (várt: false)
- Megnézem egy korrekt ip – túl rövid hossz párosra. (várt: false)
- Megnézem egy inkorrekt ip – korrekt hossz párosra. (várt: false)
- Megnézem egy inkorrekt ip – túl hosszú hossz párosra. (várt: false)
- Megnézem egy inkorrekt ip – túl rövid hossz párosra. (várt: false)
- Megnézem egy domain – korrekt hosszra (várt: false)

Megnézve a többi teszt felépítését bevezetek egy ret változót, amit kezdetben 0-ra állítok. Minden hívásnál megvizsgálom, hogy az elvárt feltétel teljesült-e. Ha nem, akkor növelem 1-gyel a ret változó értékét. A main visszatérésének a ret változót állítom be. Ha minden hiba nélkül lefutott, akkor 0-t ad vissza. Ha volt hiba, akkor azt, hogy hány darab volt.

A megírt kód hozzá:

```
#include <stdio.h>
#define __USE_XOPEN_EXTENDED
#include <string.h>
#include <faup/faup.h>
#include <faup/decode.h>
#include <faup/hash/hash.h>
#include <faup/hash/htable.h>

int main(int argc, char **argv)
{
    int ret = 0;
    if (!is_ipv4("192.168.1.12", 12)){ //correct ipv4
        ret++;
    }
    if (!is_ipv4("192.168.120.12", 14)){ //correct ipv4
        ret++;
    }
    if (is_ipv4("192.168.1.12", 15)){ //bigger length
        ret++;
    }
    if (is_ipv4("192.168.1.12", 2)){ //lower length
        ret++;
    }

    if (is_ipv4("2001:db8:a0b:12f0::1", 20)){ //ipv4
        ret++;
    }
    if (is_ipv4("2001:db8:a0b:12f0::1", 12)){
        ret++;
    }
    if (is_ipv4("2001:db8:a0b:12f0::1", 11)){
        ret++;
    }

    if (is_ipv4("http://what.com", 15)){ //domain
        ret++;
    }
    if (is_ipv4("http://what.com", 16)){ //domain
        ret++;
    }
    if (is_ipv4("http://what.com", 20)){ //domain
        ret++;
    }
    return ret;
}
```


Futtassuk le!

```
fox@Foxy: ~/faup-master/build
-- Build files have been written to: /home/fox/faup-master/build
fox@Foxy:~/faup-master/build$ make
[ 37%] Built target faup_static
[ 75%] Built target faupl
[ 81%] Built target faup
[ 84%] Built target json-tester
[ 88%] Built target hashtable
[ 92%] Built target snapshot
[ 96%] Built target issue36
Scanning dependencies of target decode-tester
[ 98%] Building C object src/tests/CMakeFiles/decode-tester.dir/decode-tester.c.o
/home/fox/faup-master/src/tests/decode-tester.c: In function 'main':
/home/fox/faup-master/src/tests/decode-tester.c:12:8: warning: implicit declaration
of function 'is_ipv4' [-Wimplicit-function-declaration]
    if (!is_ipv4("192.168.1.12", 12)){ //correct ipv4
        ^
[100%] Linking C executable decode-tester
[100%] Built target decode-tester
fox@Foxy:~/faup-master/build$ ctest
Test project /home/fox/faup-master/build
  Start 1: Url_Argument
1/12 Test #1: Url_Argument ..... Passed    0.12 sec
  Start 2: File_Argument
2/12 Test #2: File_Argument ..... Passed    0.10 sec
  Start 3: Issue24
3/12 Test #3: Issue24 ..... Passed    0.10 sec
  Start 4: Issue27
4/12 Test #4: Issue27 ..... Passed    0.10 sec
  Start 5: Issue28
5/12 Test #5: Issue28 ..... Passed    0.11 sec
  Start 6: Issue30
6/12 Test #6: Issue30 ..... Passed    0.10 sec
  Start 7: Issue69
7/12 Test #7: Issue69 ..... Passed    0.10 sec
  Start 8: Issue36
8/12 Test #8: Issue36 ..... Passed    0.01 sec
  Start 9: Snapshot
9/12 Test #9: Snapshot ..... Passed    0.00 sec
  Start 10: Hashtable
10/12 Test #10: Hashtable ..... Passed    0.00 sec
  Start 11: JsonURLsTester
11/12 Test #11: JsonURLsTester ..... Passed    0.01 sec
  Start 12: DecodeTester
12/12 Test #12: DecodeTester ..... Passed    0.00 sec

100% tests passed, 0 tests failed out of 12

Total Test time (real) =  0.76 sec
fox@Foxy:~/faup-master/build$
```

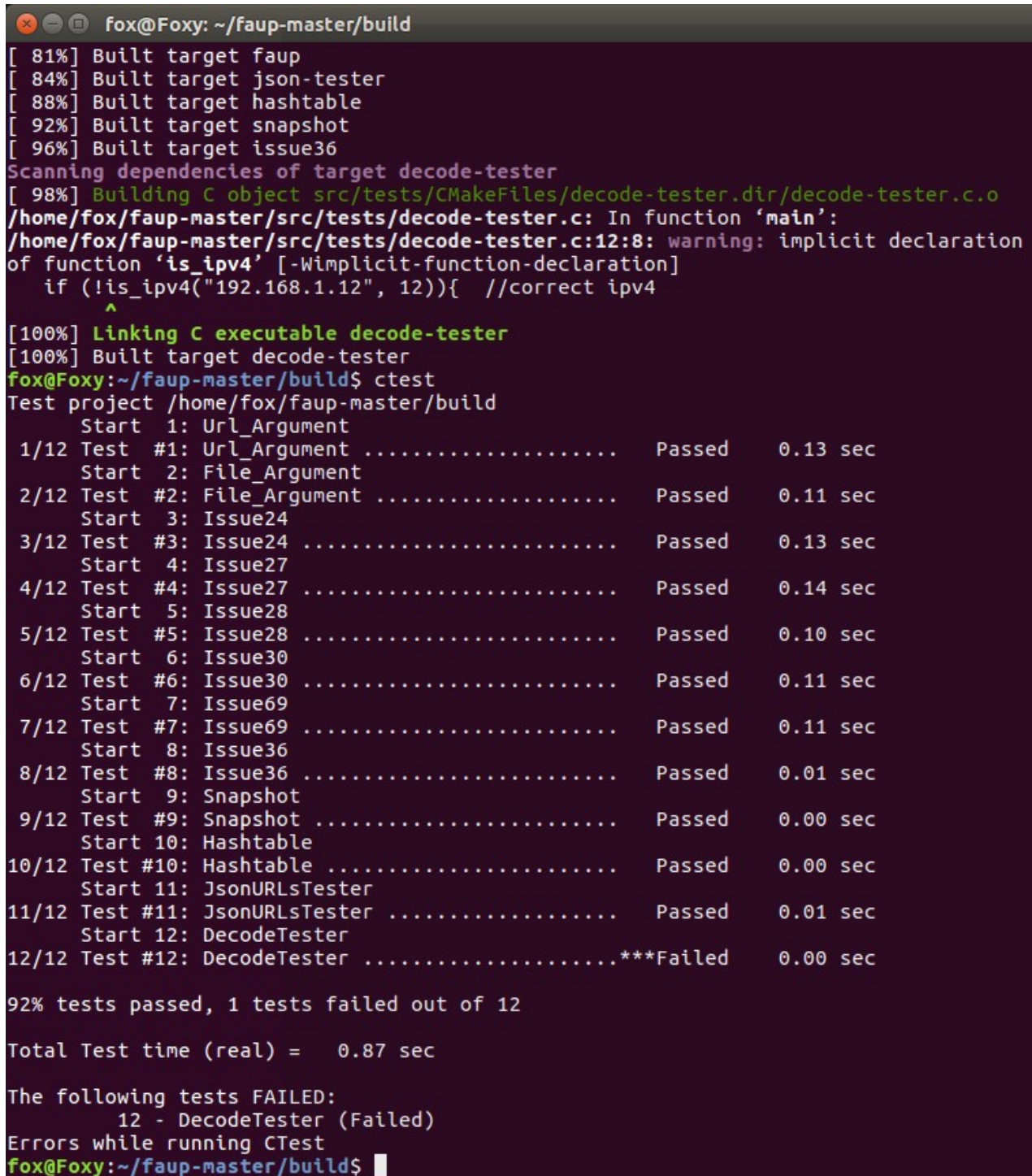
Láthatjuk, hogy lefut sikeresen.

Egy kis csalást követtem el...

Vegyük a következő kódot:

```
if (is_ipv4("192.168.1.12", 10)){
    ret++;
}
```

Ennek false-szal kellene visszatérni, ám ha így lefuttatjuk a tesztet, akkor az failed állapotba ér:



```
fox@Foxy: ~/faup-master/build
[ 81%] Built target faup
[ 84%] Built target json-tester
[ 88%] Built target hashtable
[ 92%] Built target snapshot
[ 96%] Built target issue36
Scanning dependencies of target decode-tester
[ 98%] Building C object src/tests/CMakeFiles/decode-tester.dir/decode-tester.c.o
/home/fox/faup-master/src/tests/decode-tester.c: In function 'main':
/home/fox/faup-master/src/tests/decode-tester.c:12:8: warning: implicit declaration
of function 'is_ipv4' [-Wimplicit-function-declaration]
    if (!is_ipv4("192.168.1.12", 12)){ //correct ipv4
        ^
[100%] Linking C executable decode-tester
[100%] Built target decode-tester
fox@Foxy:~/faup-master/build$ ctest
Test project /home/fox/faup-master/build
   Start 1: Url_Argument
 1/12 Test #1: Url_Argument ..... Passed    0.13 sec
   Start 2: File_Argument
 2/12 Test #2: File_Argument ..... Passed    0.11 sec
   Start 3: Issue24
 3/12 Test #3: Issue24 ..... Passed    0.13 sec
   Start 4: Issue27
 4/12 Test #4: Issue27 ..... Passed    0.14 sec
   Start 5: Issue28
 5/12 Test #5: Issue28 ..... Passed    0.10 sec
   Start 6: Issue30
 6/12 Test #6: Issue30 ..... Passed    0.11 sec
   Start 7: Issue69
 7/12 Test #7: Issue69 ..... Passed    0.11 sec
   Start 8: Issue36
 8/12 Test #8: Issue36 ..... Passed    0.01 sec
   Start 9: Snapshot
 9/12 Test #9: Snapshot ..... Passed    0.00 sec
   Start 10: Hashtable
10/12 Test #10: Hashtable ..... Passed    0.00 sec
   Start 11: JsonURLsTester
11/12 Test #11: JsonURLsTester ..... Passed    0.01 sec
   Start 12: DecodeTester
12/12 Test #12: DecodeTester .....***Failed    0.00 sec

92% tests passed, 1 tests failed out of 12

Total Test time (real) =  0.87 sec

The following tests FAILED:
    12 - DecodeTester (Failed)
Errors while running CTest
fox@Foxy:~/faup-master/build$
```

Értékelés: ez egy logikai hiba! Ilyenkor élesben nyitok róla egy issue ticketet, hogy a fejlesztők ki tudják javítani. Ez egy jó példa rá, hogy miért érdemes tesztet írni