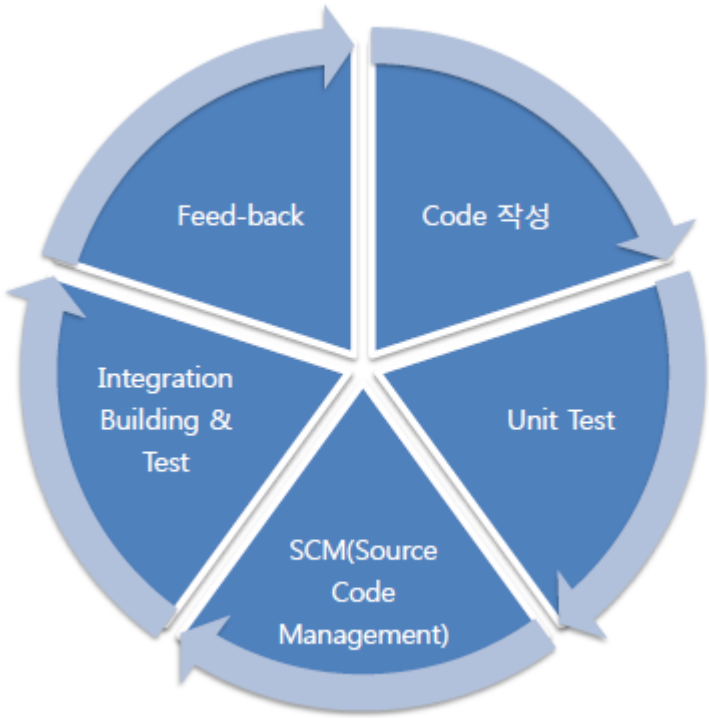
CTIP for C
& Unit Test

200511305 김성규
200511306 김성훈
200614164 김효석
200611124 유성배
200518036 곡진화

Establish CTIP Enviroment

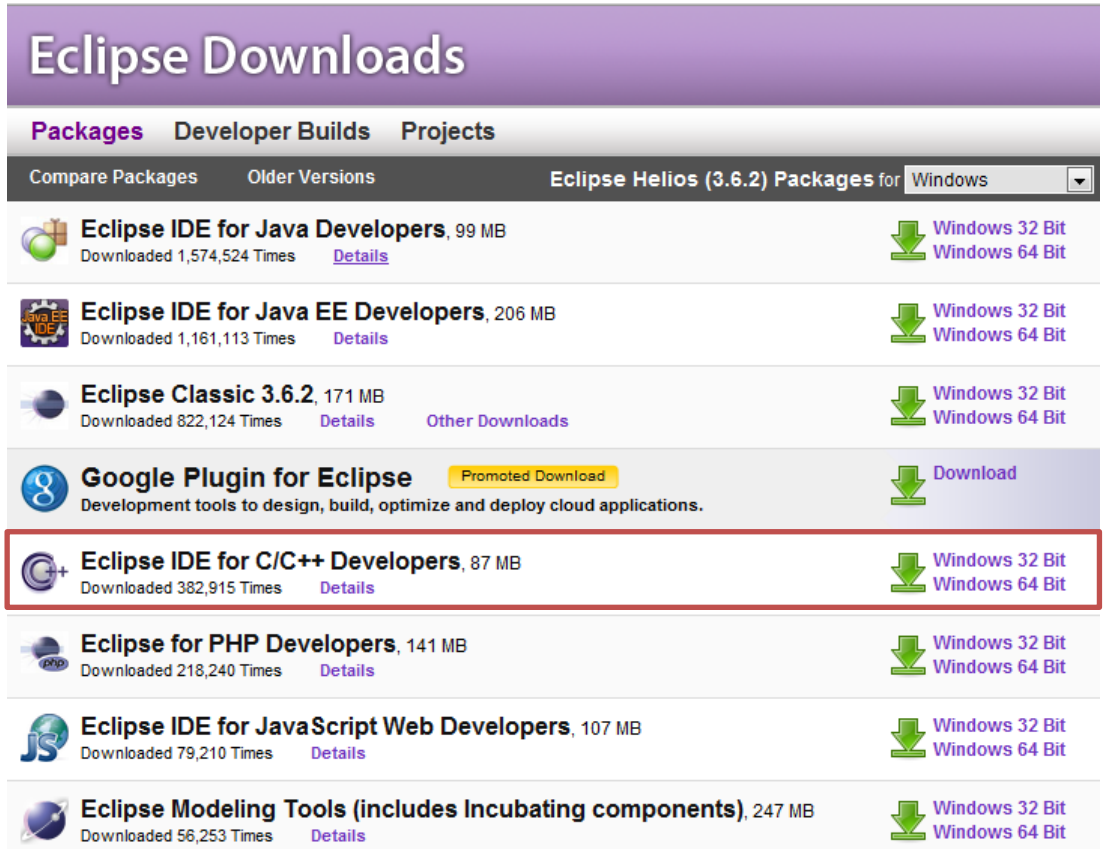
CTIP

- CTIP
  - Continuous Test & Integration Platform
  - Code Editor
    - Eclipse CDT
  - Unit Test
    - CUnit
  - Source Code Management
    - Subclipse
  - Integration Building
    - make command & makefile
  - CI Tool
    - Hudson



Code Editor

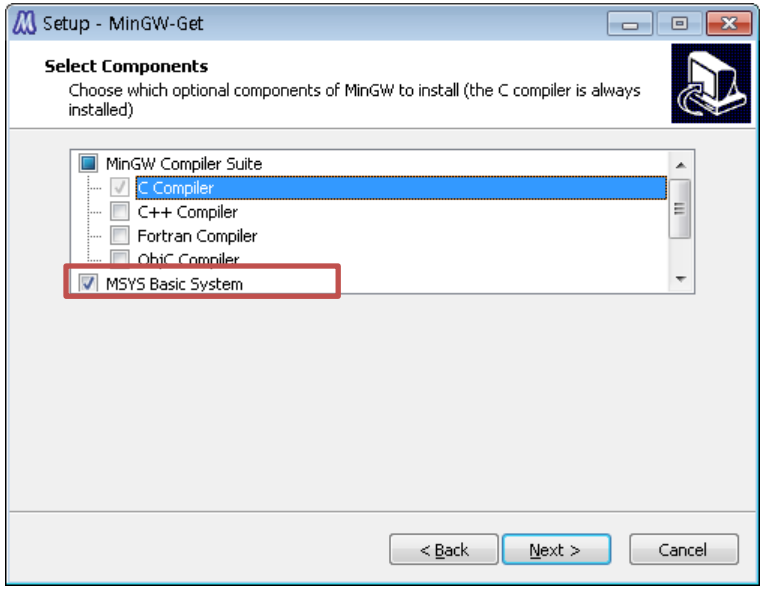
- Ecliplse IDE for C/C++ Developers
  - http://www.eclipse.org/downloads/



But Eclipse not support Compiler!!

Code Editor

- C Comilper for Windows
  - Mingw(Minimalist GNU for Windows)
  - http://sourceforge.net/projects/mingw/

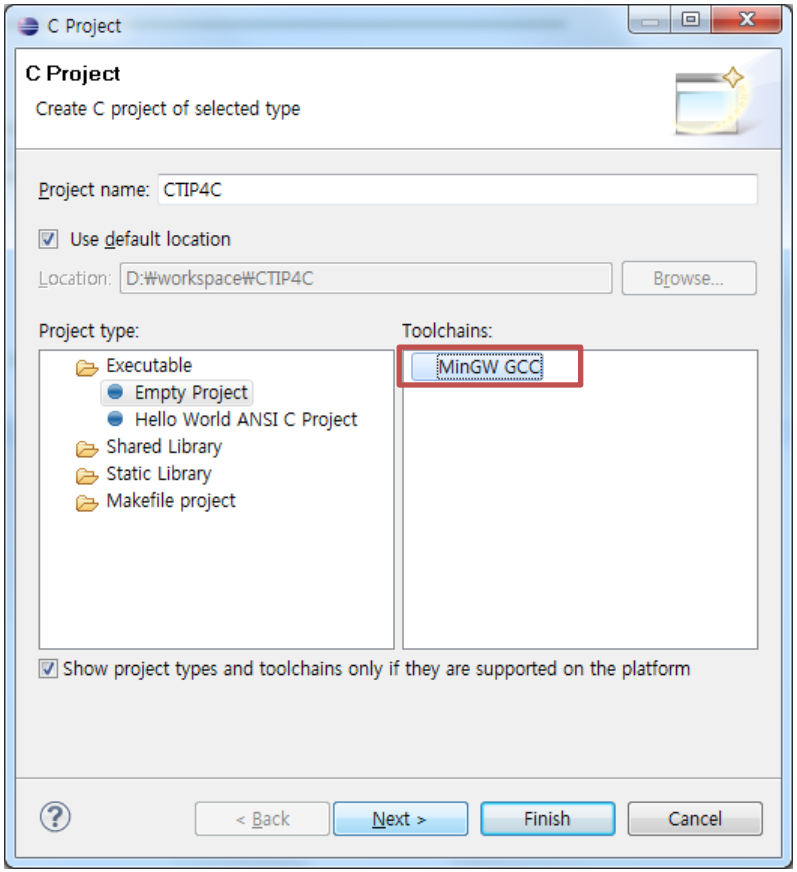


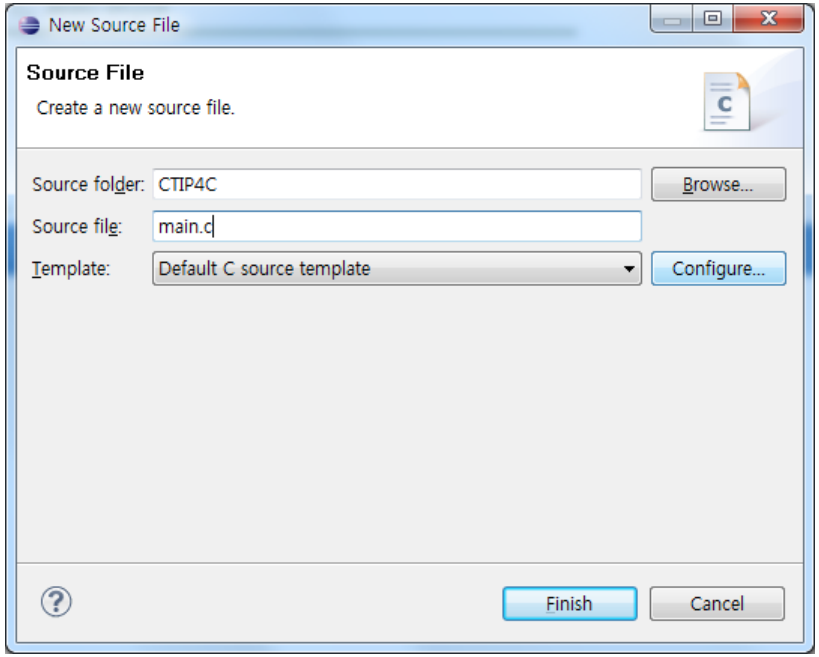
MSYS(Minimal System)
- Provide shell environment
- Install for installing CUnit

  - Adding Mingw's bin folder to Path enviroment variables
    - C:₩MinGW₩bin
    - C:₩MinGW₩msys₩1.0₩bin

Code Editor

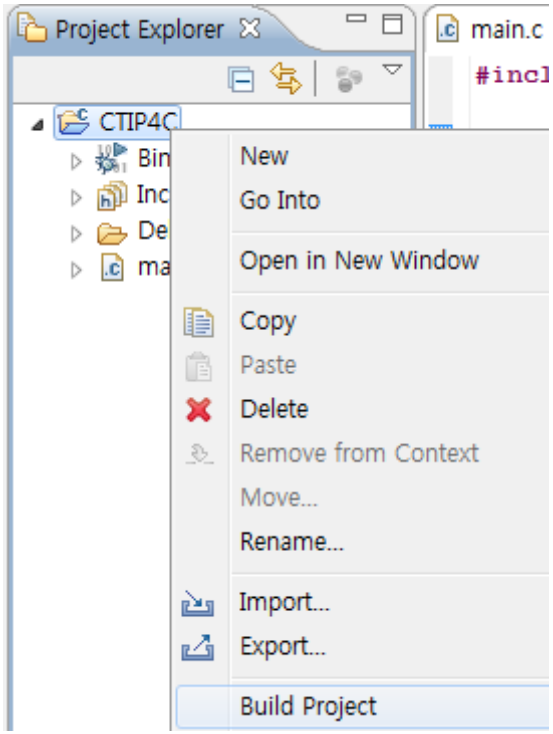
- Create Project, file
  - File>New>C Project, Source File

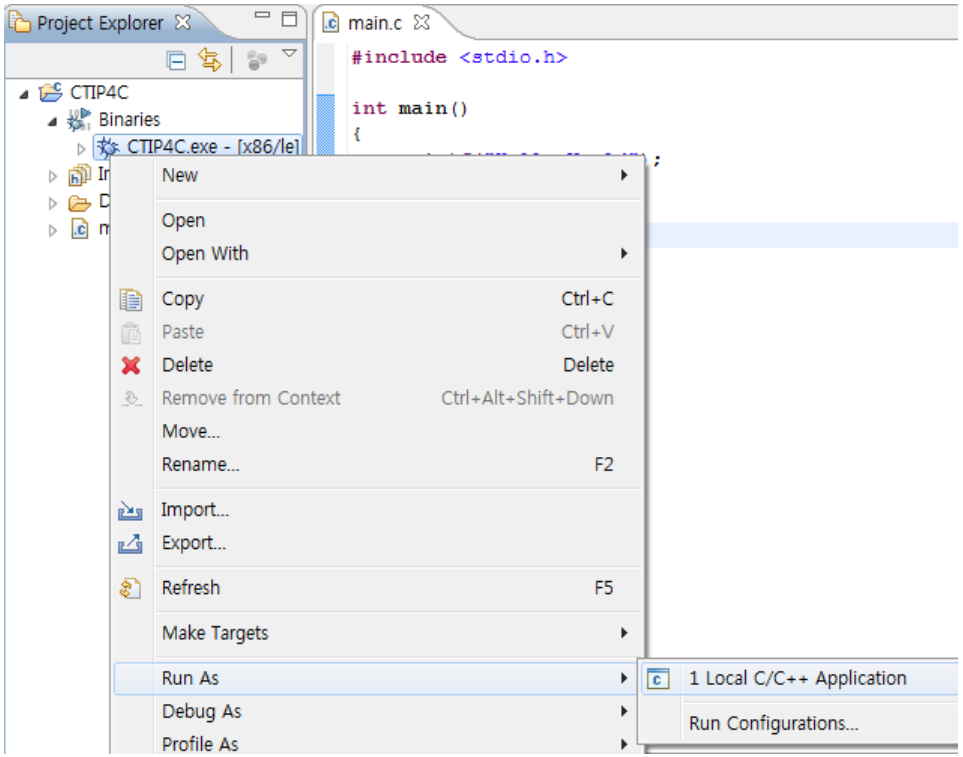




Code Editor

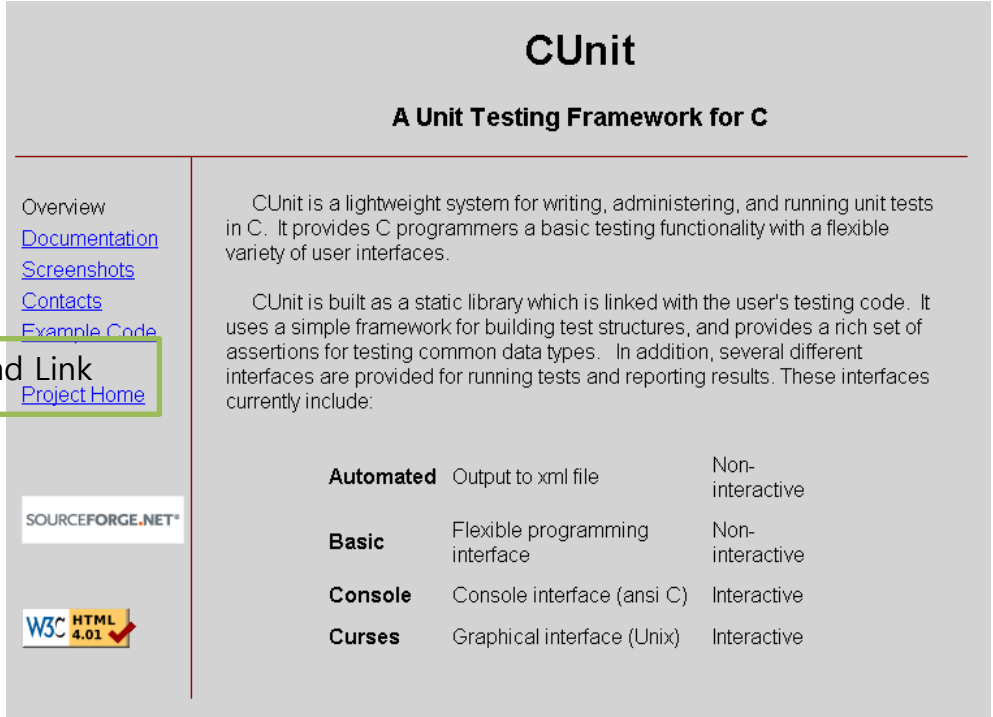
❖ Build / Run





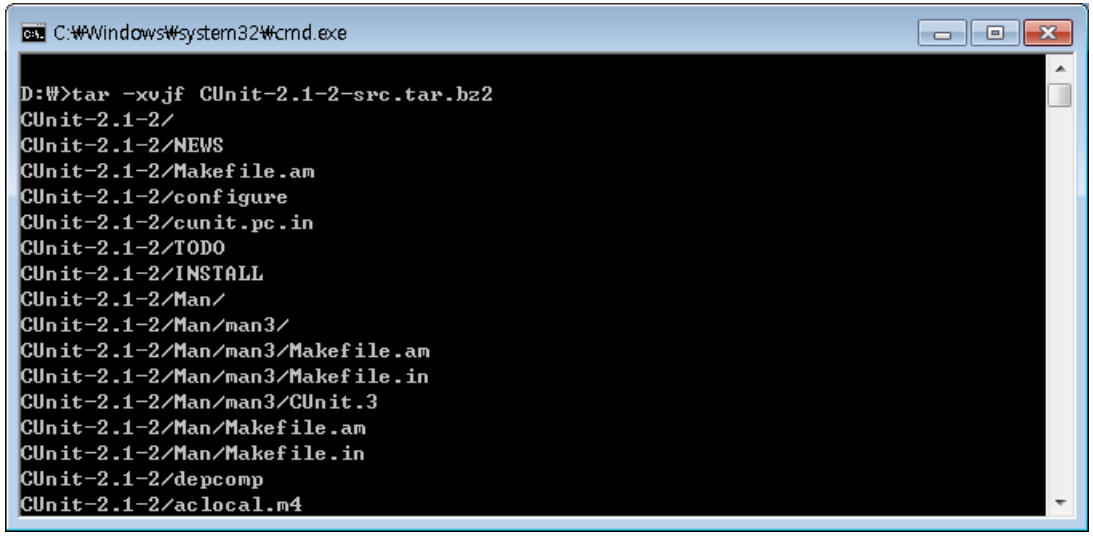
Unit Test

- CUnit
  - Unit Testing Framework for C
  - Platform dependent (Curses on Unix)
  - http://cunit.sourceforge.net/



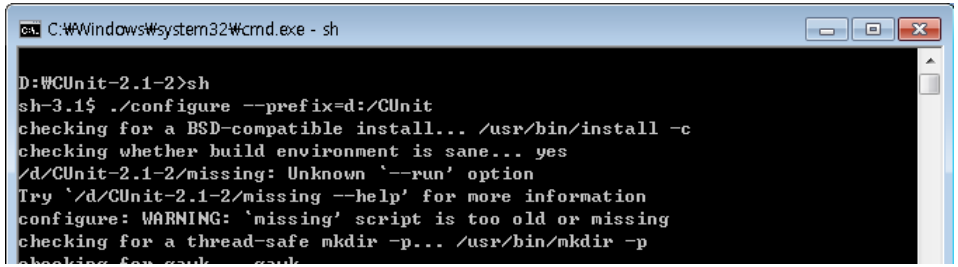
Unit Test

- Install CUnit(1/3)
  - Download “CUnit-2.1-2-src.tar.bz2”
    - http://sourceforge.net/projects/cunit/
  - Decompression File
    - tar -xvjf CUnit-2.1-2-src.tar.bz2

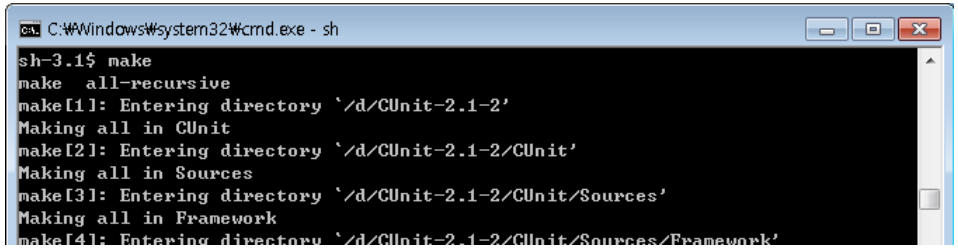


Unit Test

- Install CUnit(2/3)
  - shell mode
    - sh (MSYS command)
  - set Install configure
    - ./configure --prefix=<Install path>

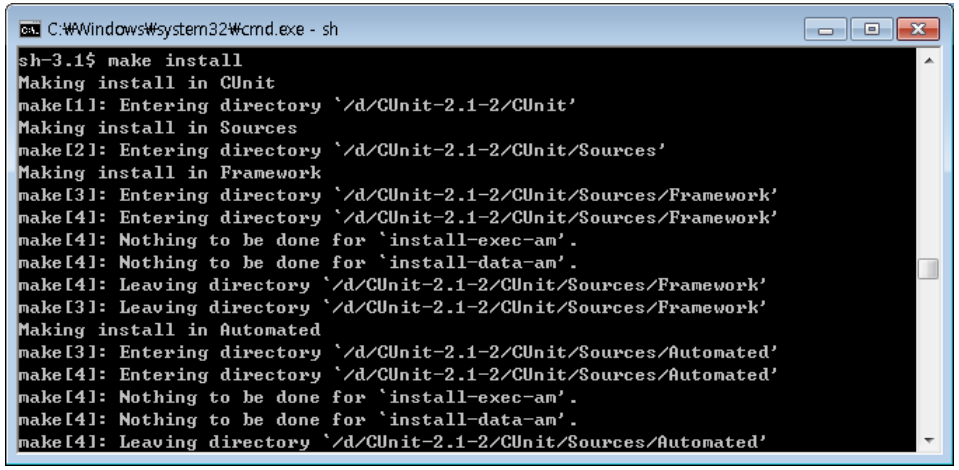


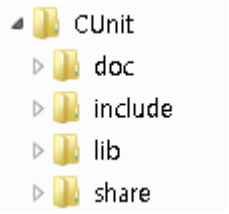
  - Generate Install makefile by configure
    - make



Unit Test

- Install CUnit(3/3)
  - Install CUnit
    - make install

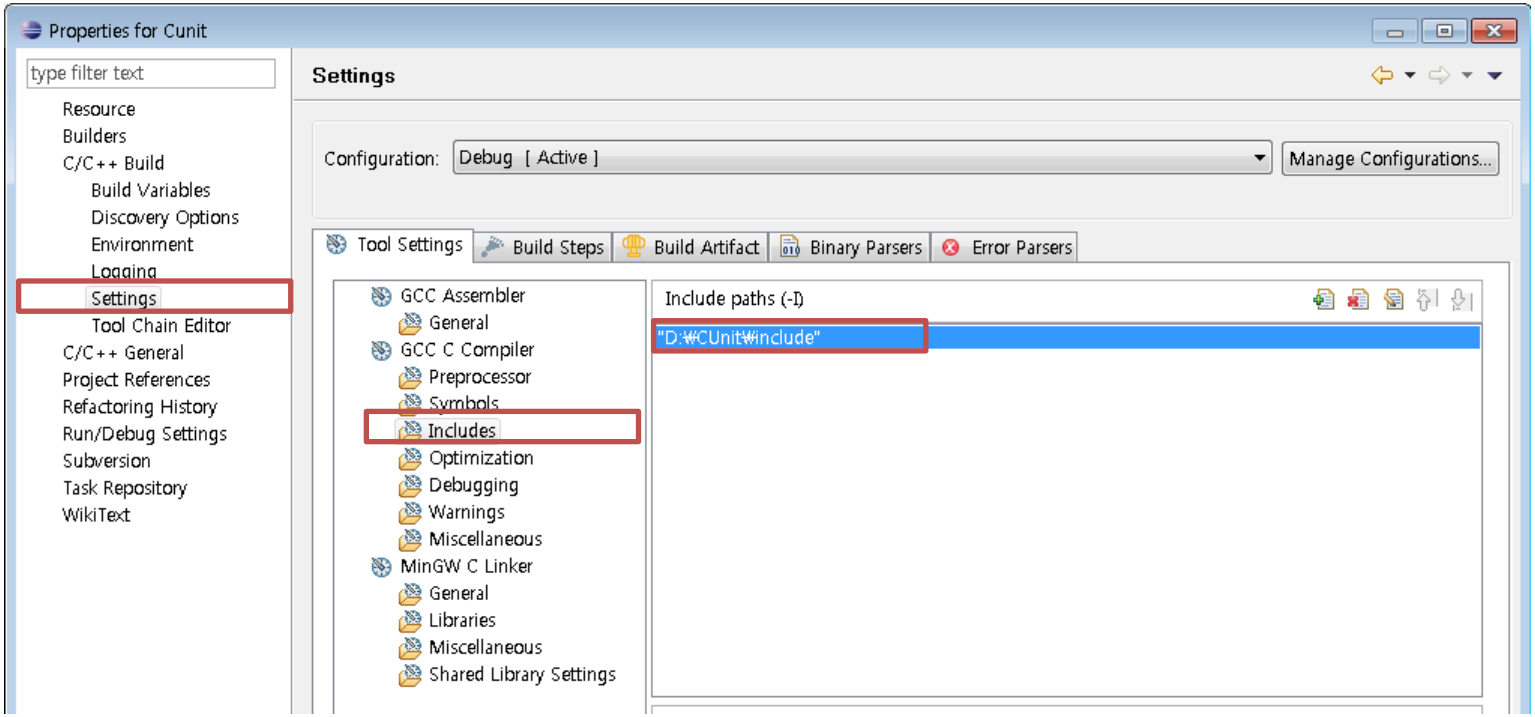




- doc : CUnit document(Html)
- include : Header files
- lib : Libraries
- share : XML DTD, XSL, man files

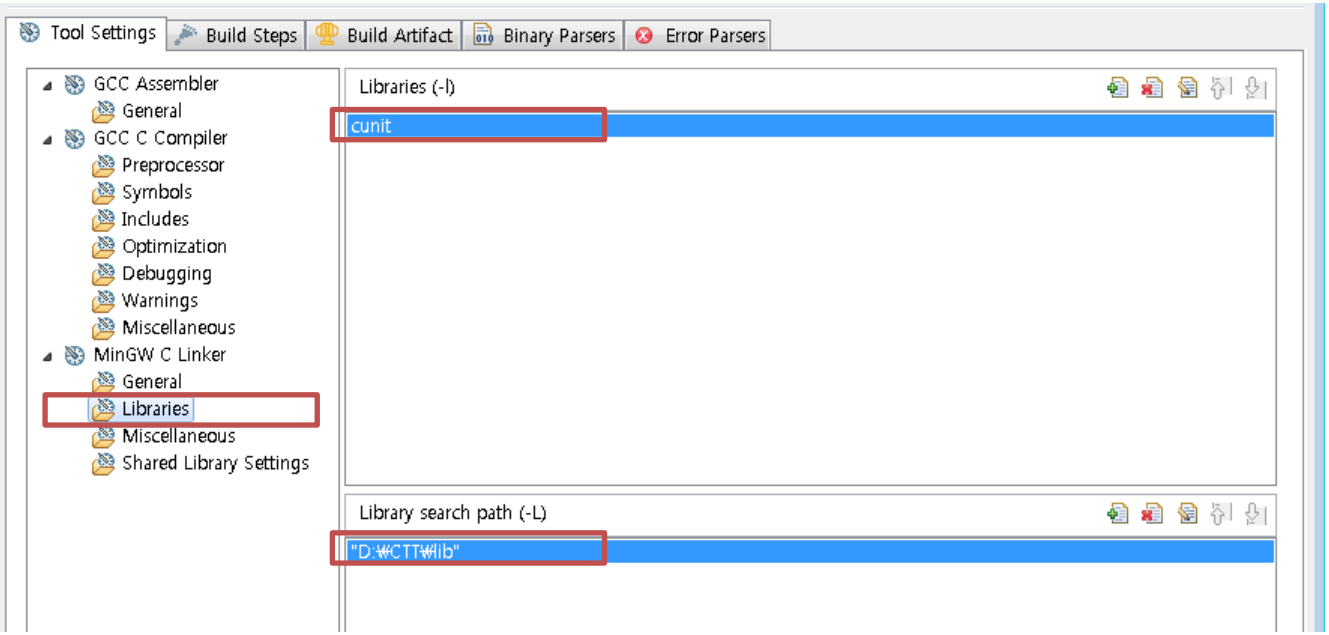
Unit Test

- Setting for Eclipse
  - Setting Compiler's Include path
    - Project>Properties>C/C++Build>Settings
    - GCC C Compiler>Includes
    - add CUnit's include folder



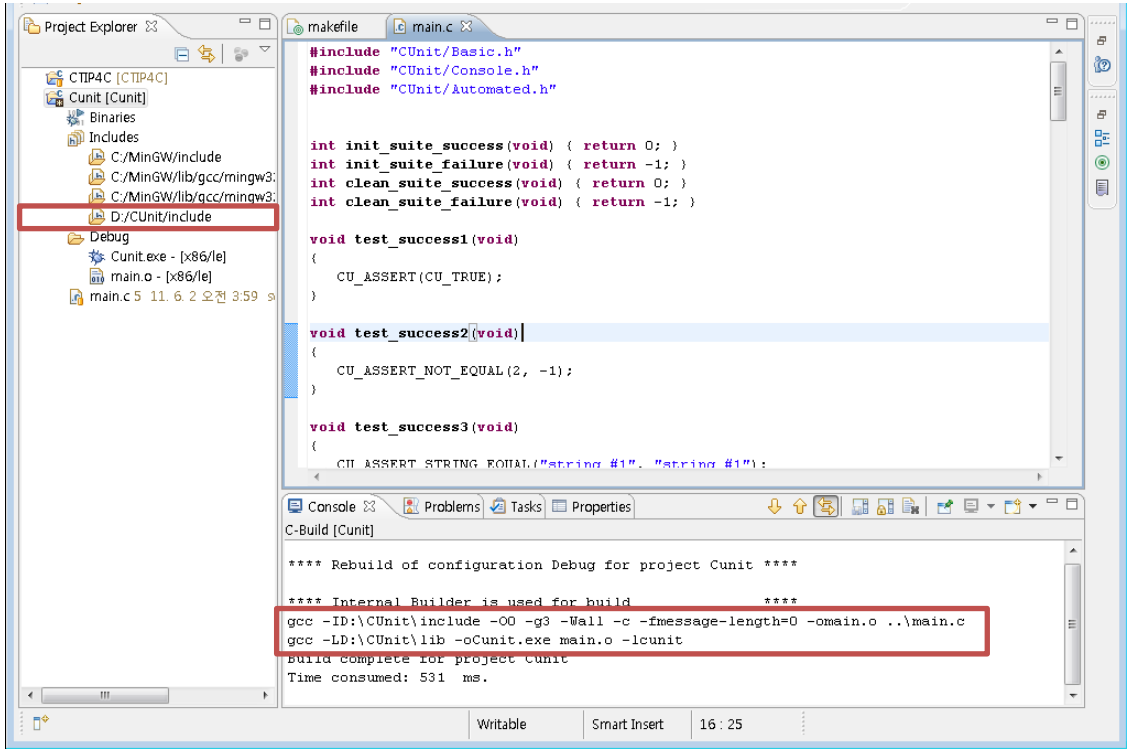
Unit Test

- Setting for Eclipse
  - Setting Linker's Libraries
    - Project>Properties>C/C++Build>Settings
    - MinGW C Linker>Libraries
    - add Libraries "cunit"
    - add CUnit's lib folder



Unit Test

- Test CUnit(1/3)
  - Using CUnit's Example Code
    - Build



Unit Test

❖ Test CUnit(2/3)

■ Run Executable file

- Using Basic.h

```
     CUnit - A unit testing framework for C - Version 2.1-2
     http://cunit.sourceforge.net/


Suite: Suite_success
  Test: successful_test_1 ...passed
  Test: successful_test_2 ...passed
  Test: successful_test_3 ...passed
WARNING - Suite initialization failed for 'Suite_init_failure'.
Suite: Suite_clean_failure
  Test: successful_test_4 ...passed
  Test: failed_test_2 ...FAILED
    1. ..\main.c:38  - CU_ASSERT_EQUAL(2,3)
  Test: successful_test_1 ...passed
WARNING - Suite cleanup failed for 'Suite_clean_failure'.
Suite: Suite_mixed
  Test: successful_test_2 ...passed
  Test: failed_test_4 ...FAILED
    1. ..\main.c:48  - CU_ASSERT_STRING_EQUAL("string #1","string #2")
  Test: failed_test_2 ...FAILED
    1. ..\main.c:38  - CU_ASSERT_EQUAL(2,3)
  Test: successful_test_4 ...passed

Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      4      3    n/a      2        0
               tests     13     10      7      3        0
             asserts     10     10      7      3      n/a

Elapsed time =    0.001 seconds


  1. CUnit System:0  - Suite Initialization failed - Suite Skipped
  2. ..\main.c:38  - CU_ASSERT_EQUAL(2,3)
  3. CUnit System:0  - Suite cleanup failed.
  4. ..\main.c:48  - CU_ASSERT_STRING_EQUAL("string #1","string #2")
  5. ..\main.c:38  - CU_ASSERT_EQUAL(2,3)
```

- Using Console.h

```
     CUnit - A Unit testing framework for C - Version 2.1-2
             http://cunit.sourceforge.net/


***************** CUNIT CONSOLE - MAIN MENU ****************************
(R)un  (S)elect  (L)ist  (A)ctivate  (F)ailures  (O)ptions  (H)elp  (Q)uit
Enter command:
```

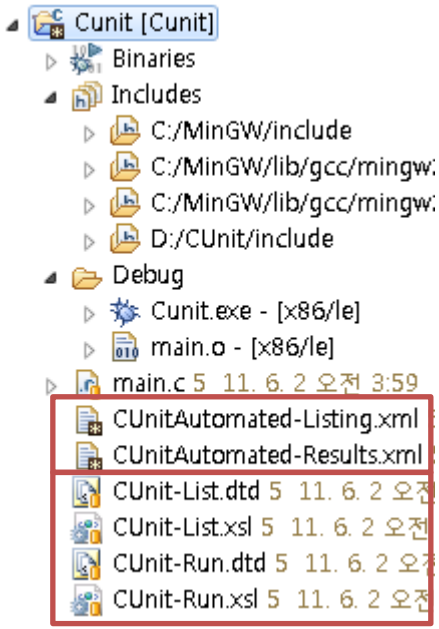
```
***************** CUNIT CONSOLE - MAIN MENU ****************************
(R)un  (S)elect  (L)ist  (A)ctivate  (F)ailures  (O)ptions  (H)elp  (Q)uit
Enter command: F

--------------- Test Run Failures -------------------------
   src_file:line# : (suite:test) : failure_condition

1. CUnit System:0 : (Suite_init_failure : ) : Suite Initialization failed -
2. ..\main.c:38 : (Suite_clean_failure : failed_test_2) : CU_ASSERT_EQUAL(2,
3. CUnit System:0 : (Suite_clean_failure : ) : Suite cleanup failed.
4. ..\main.c:48 : (Suite_mixed : failed_test_4) : CU_ASSERT_STRING_EQUAL("st
5. ..\main.c:38 : (Suite_mixed : failed_test_2) : CU_ASSERT_EQUAL(2,3)
-----------------------------------------------------------
Total Number of Failures : 5
```

Unit Test

- Test CUnit(3/3)
  - Run Executable file
    - Using Automated.h



- Executable file generate XML file

- Use CUnit/share's DTD, XSL files

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="CUnit-Run.xsl" ?>
<!DOCTYPE CUNIT_TEST_RUN_REPORT SYSTEM "CUnit-Run.dtd">
<CUNIT_TEST_RUN_REPORT>
  <CUNIT_HEADER/>
  <CUNIT_RESULT_LISTING>
    <CUNIT_RUN_SUITE>
      <CUNIT_RUN_SUITE_SUCCESS>
        <SUITE_NAME> Suite_success </SUITE_NAME>
        <CUNIT_RUN_TEST_RECORD>
          <CUNIT_RUN_TEST_SUCCESS>
            <TEST_NAME> successful_test_1 </TEST_NAME>
          </CUNIT_RUN_TEST_SUCCESS>
        </CUNIT_RUN_TEST_RECORD>
        <CUNIT_RUN_TEST_RECORD>
          <CUNIT_RUN_TEST_SUCCESS>
            <TEST_NAME> successful_test_2 </TEST_NAME>
          </CUNIT_RUN_TEST_SUCCESS>
        </CUNIT_RUN_TEST_RECORD>
        <CUNIT_RUN_TEST_RECORD>
          <CUNIT_RUN_TEST_SUCCESS>
            <TEST_NAME> successful_test_3 </TEST_NAME>
          </CUNIT_RUN_TEST_SUCCESS>
        </CUNIT_RUN_TEST_RECORD>
      </CUNIT_RUN_SUITE_SUCCESS>
    </CUNIT_RUN_SUITE>
```

Automated Test Run Results

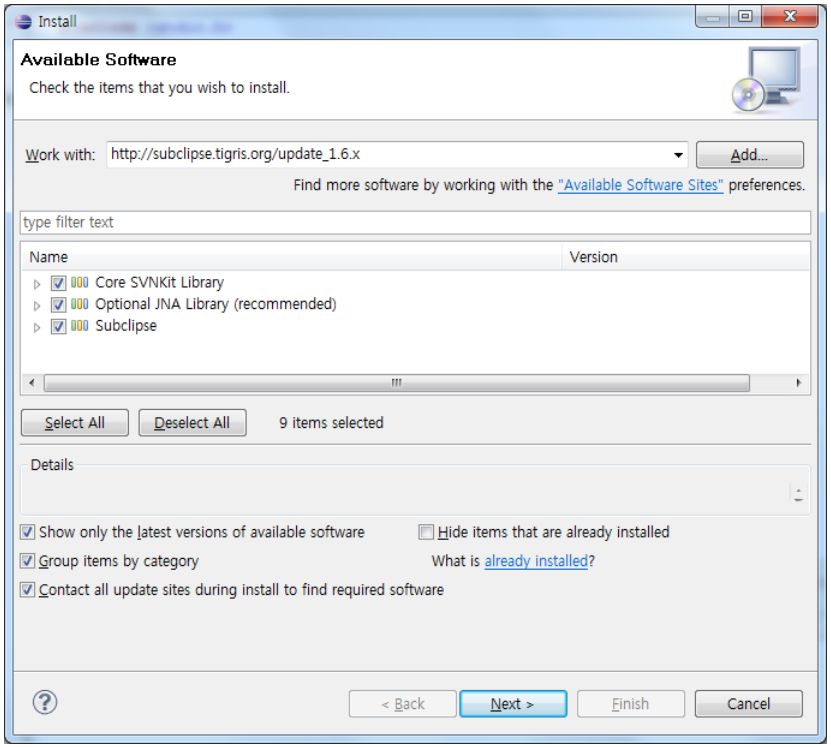
Running Suite Suite_success		
	Running test successful_test_1 ...	Passed
	Running test successful_test_2 ...	Passed
	Running test successful_test_3 ...	Passed
Running Suite Suite_init_failure ...		Suite Initialization Failed
Running Suite Suite_clean_failure		
	Running test successful_test_4 ...	Passed
	Running test failed_test_2 ...	Failed
File Name	..\main.c	**Line Number** 38
Condition	CU_ASSERT_EQUAL(2,3)	
	Running test successful_test_1 ...	Passed
Running Suite Suite_clean_failure ...		Suite Cleanup Failed

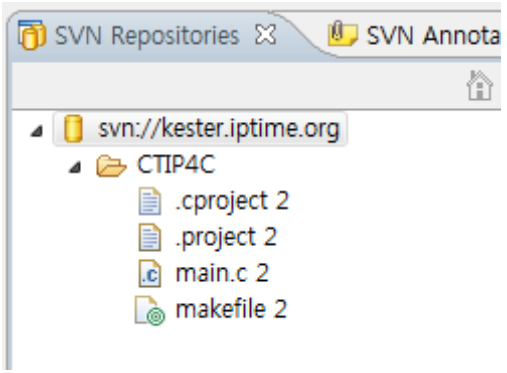
Listing of Registered Suites & Tests

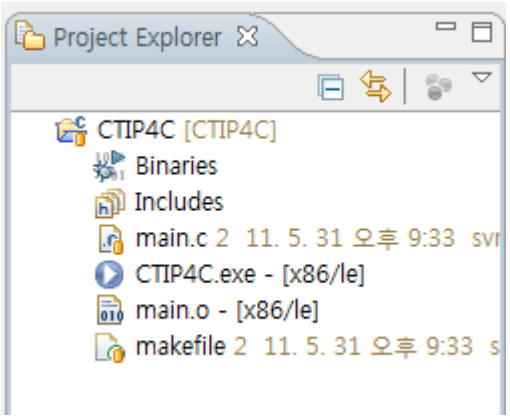
		Initialize Function?	Cleanup Function?	Test Count	Active?
Suite	**Suite_success**	Yes	Yes	3	Yes
Test	successful_test_1				Yes
Test	successful_test_2				Yes
Test	successful_test_3				Yes
Suite	**Suite_init_failure**	Yes	No	3	Yes
Test	successful_test_1				Yes
Test	successful_test_2				Yes
Test	successful_test_3				Yes
Suite	**Suite_clean_failure**	No	Yes	3	Yes
Test	successful_test_4				Yes
Test	failed_test_2				Yes
Test	successful_test_1				Yes

SCM

- Source Code Management
  - Subversion - Subclipse
  - Help>Install New Software
  - http://subclipse.tigris.org/update_1.6.x

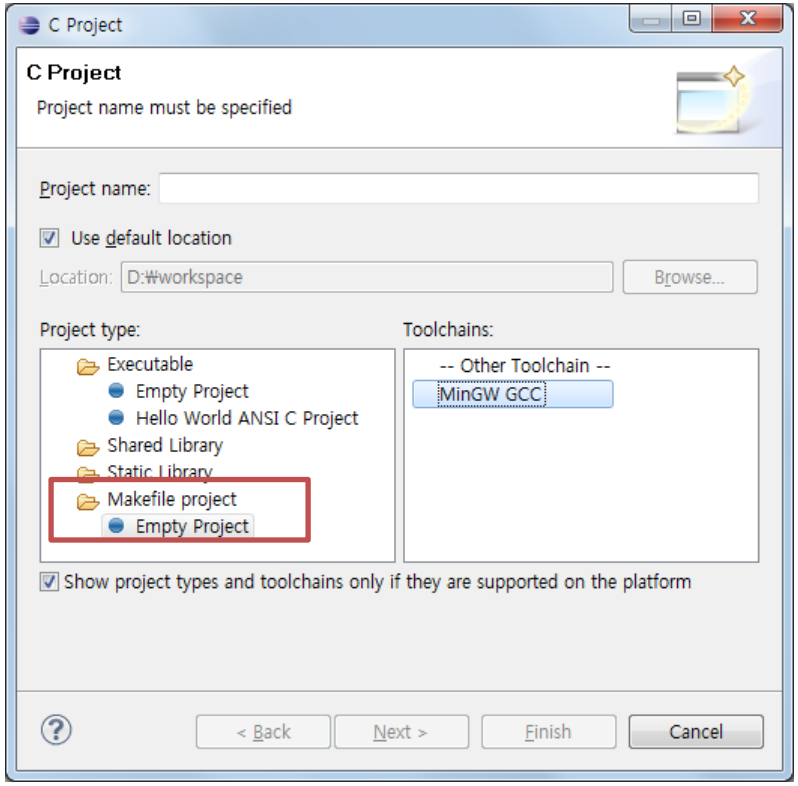


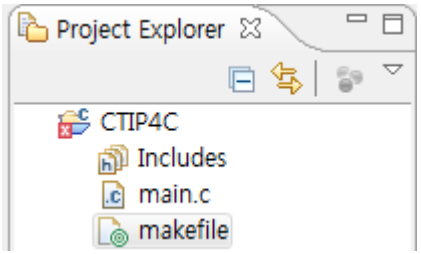


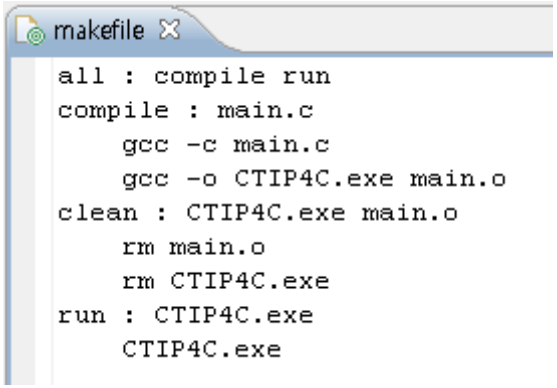


Integration Building

- Use makefile for Integration building
  - similar to build script
  - Create Makefile Project, makefile
    - New>C Project, File

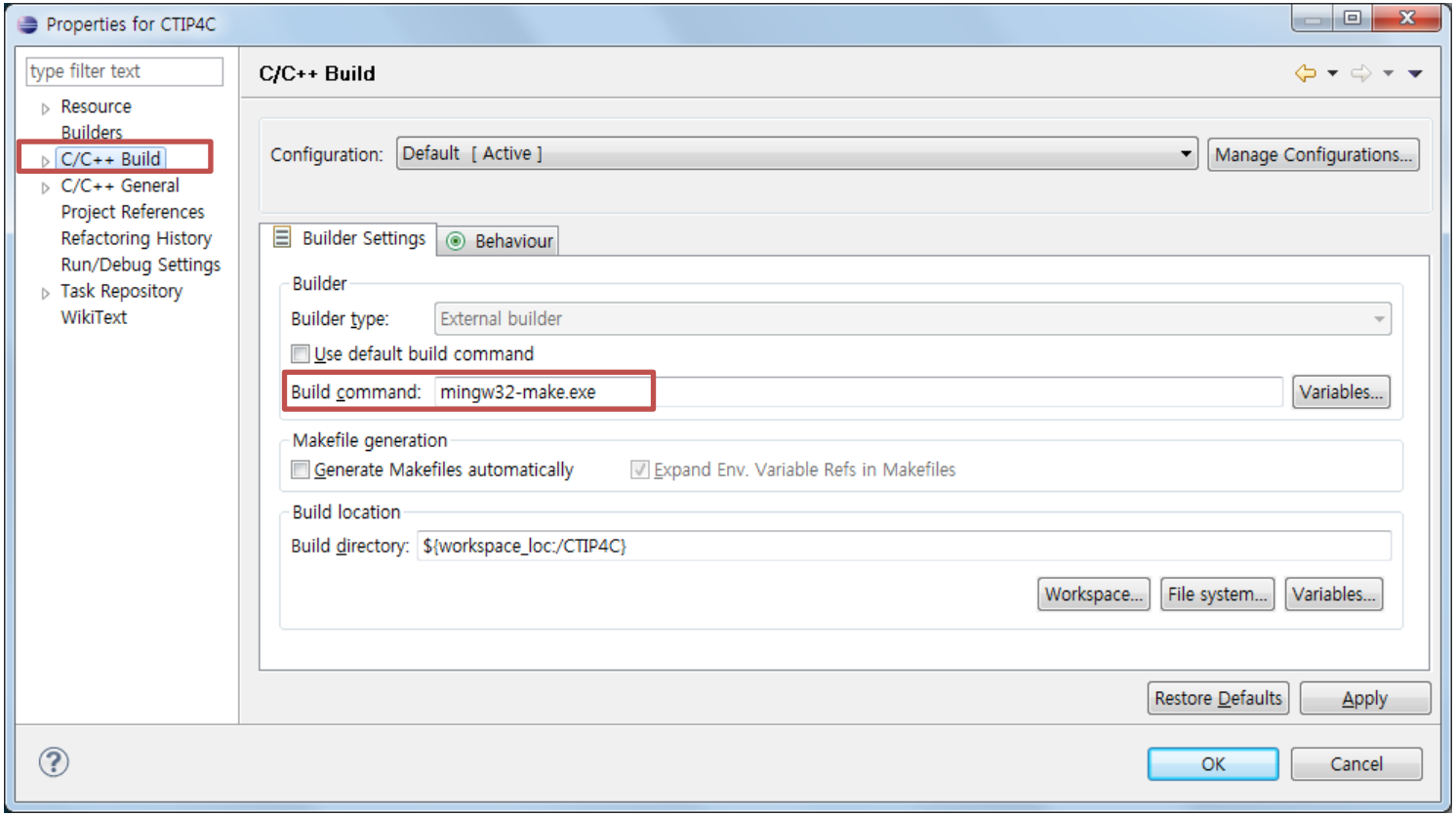






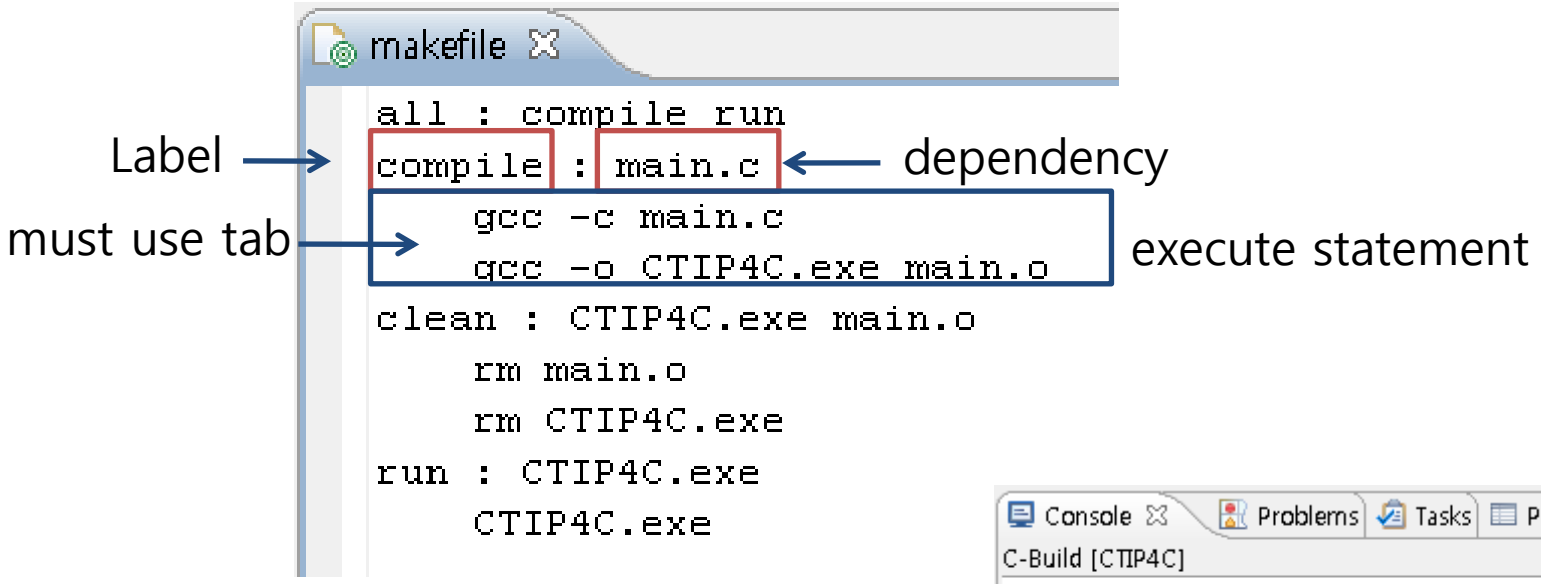
Integration Building

- Use mingw32-make.exe
  - Proejct>Properties>C/C++ Build>Builder Settings
  - Set Build command : mingw32-make.exe



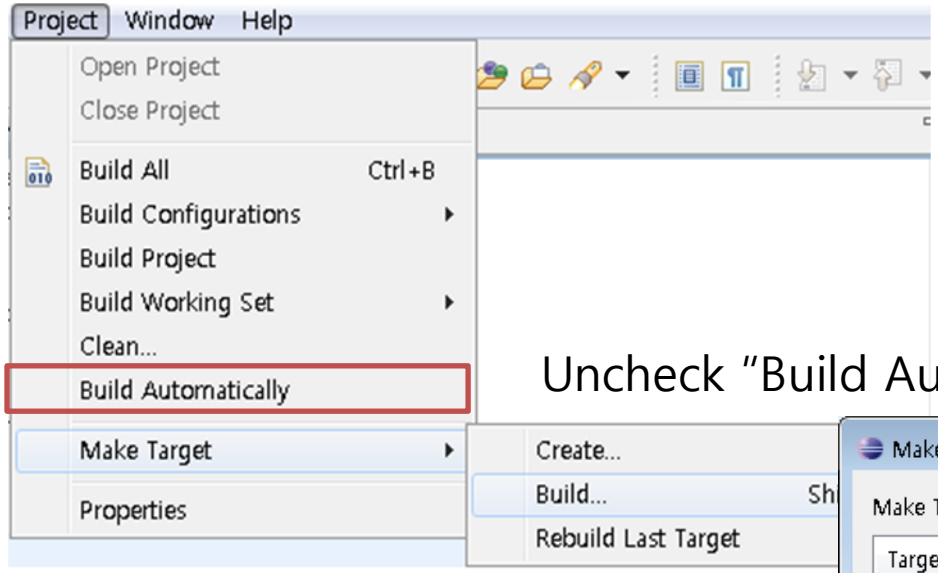
Integration Building

- Edit makefile
  - Build Defualt : Label “all”



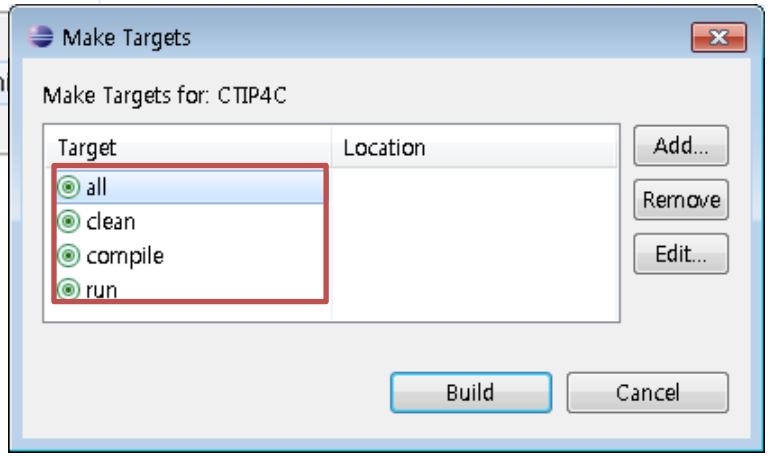
Integration Building

- Selective Build
  - Project>Make Target>Build



Uncheck "Build Automatically"

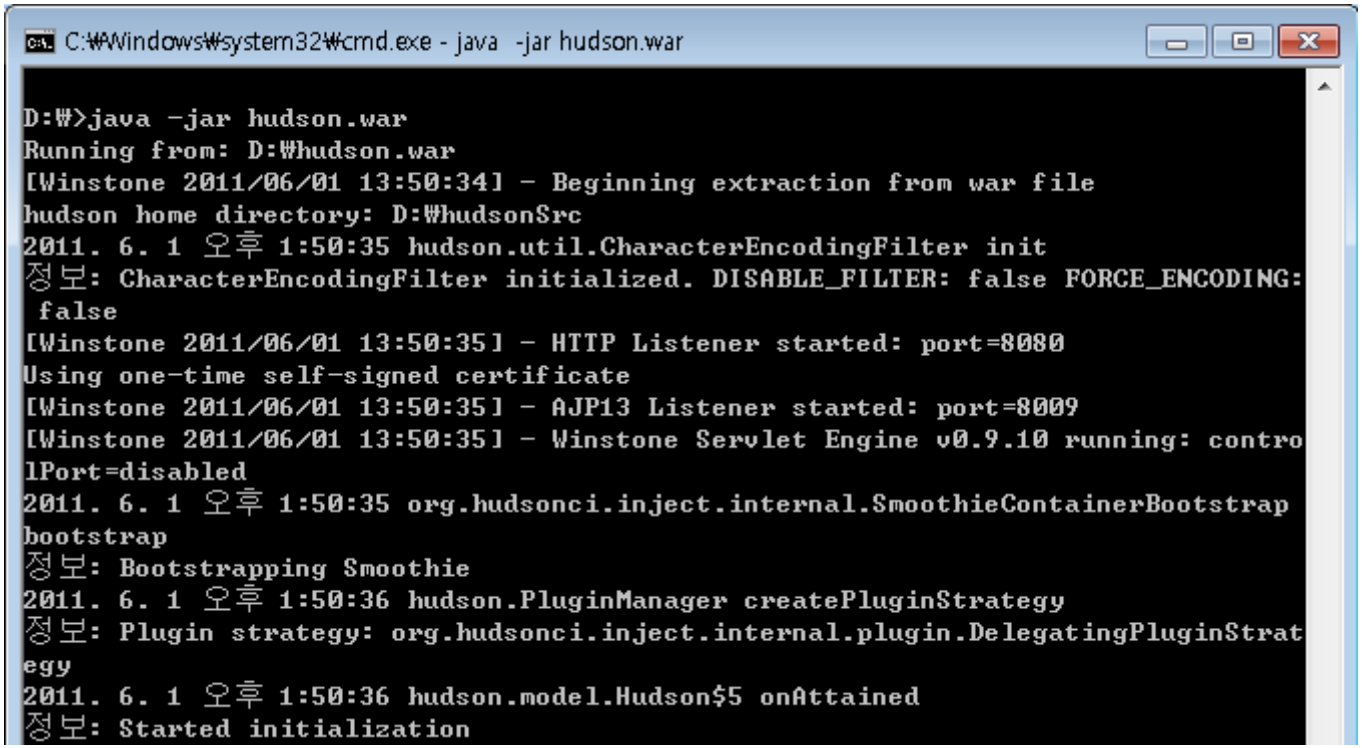
Target is makefile's label



Selective Build in Command-line
: mingw32-make <label>

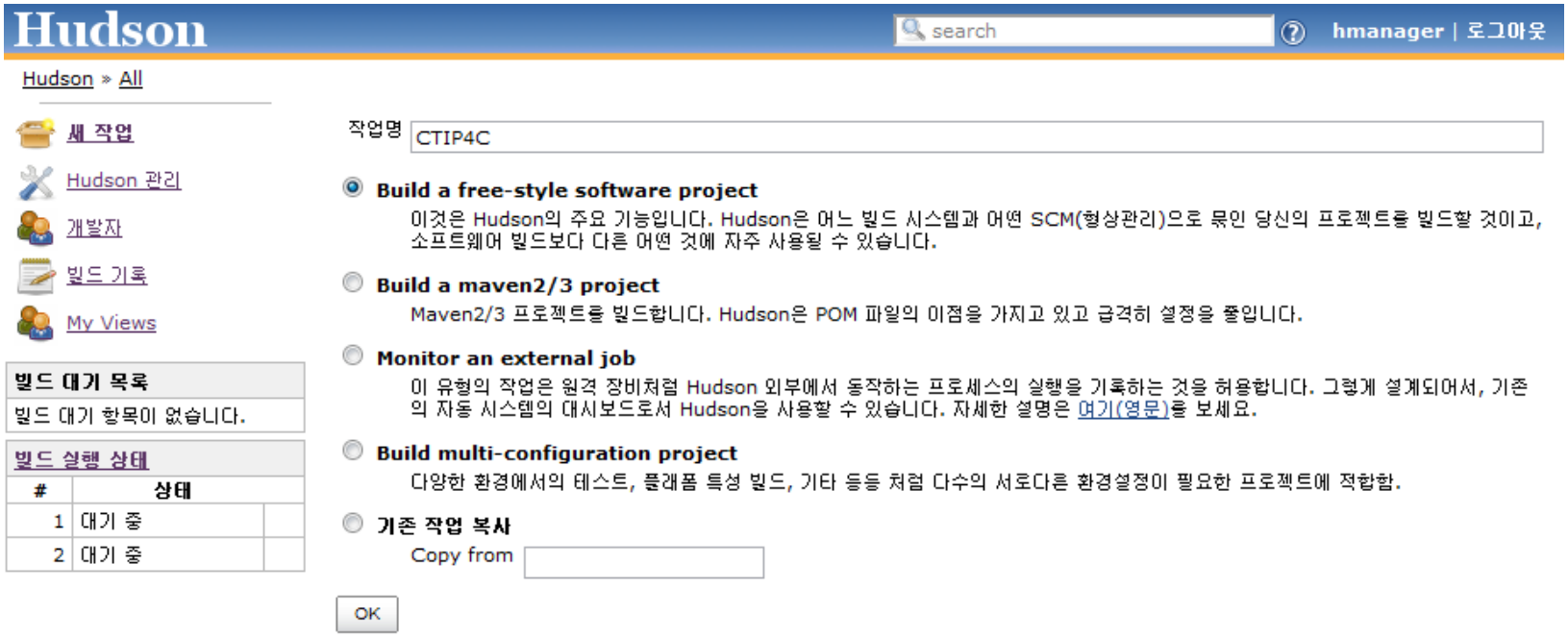
CI Server

- Hudson
  - http://hudson-ci.org/
  - Download “hudson.war”
  - java -jar hudson.war



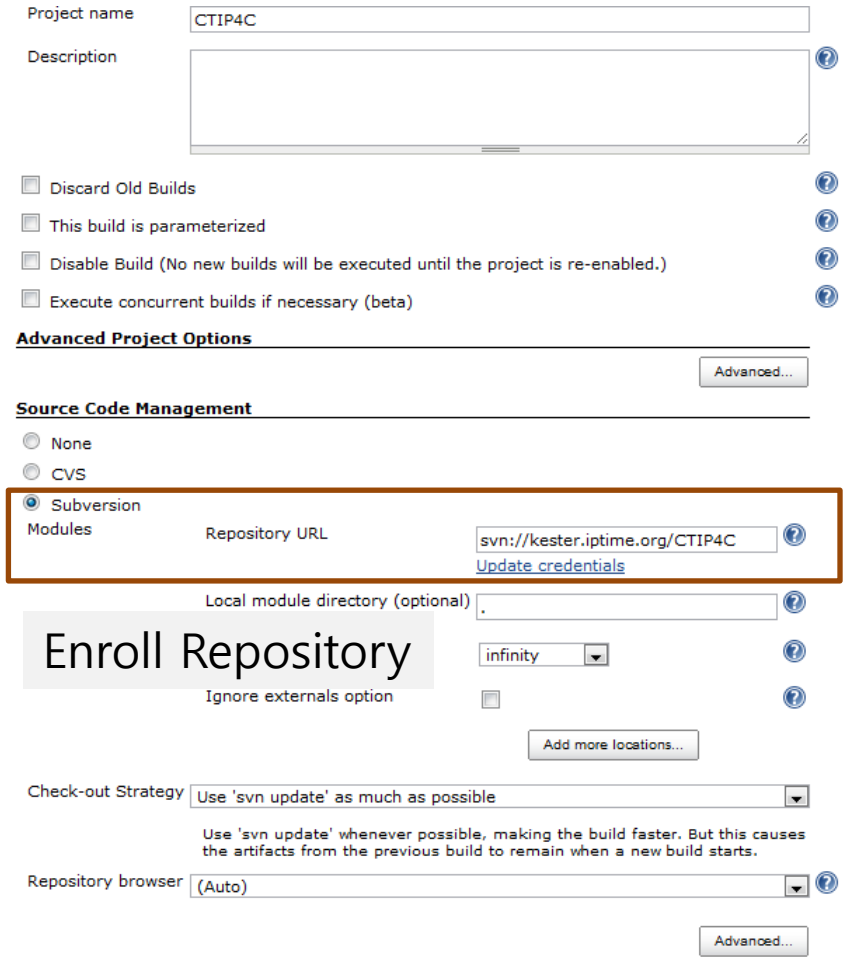
CI Server

- Hudson
  - http://<ip address>:8080
  - Create New Job
  - Select “Build a free-style software project”

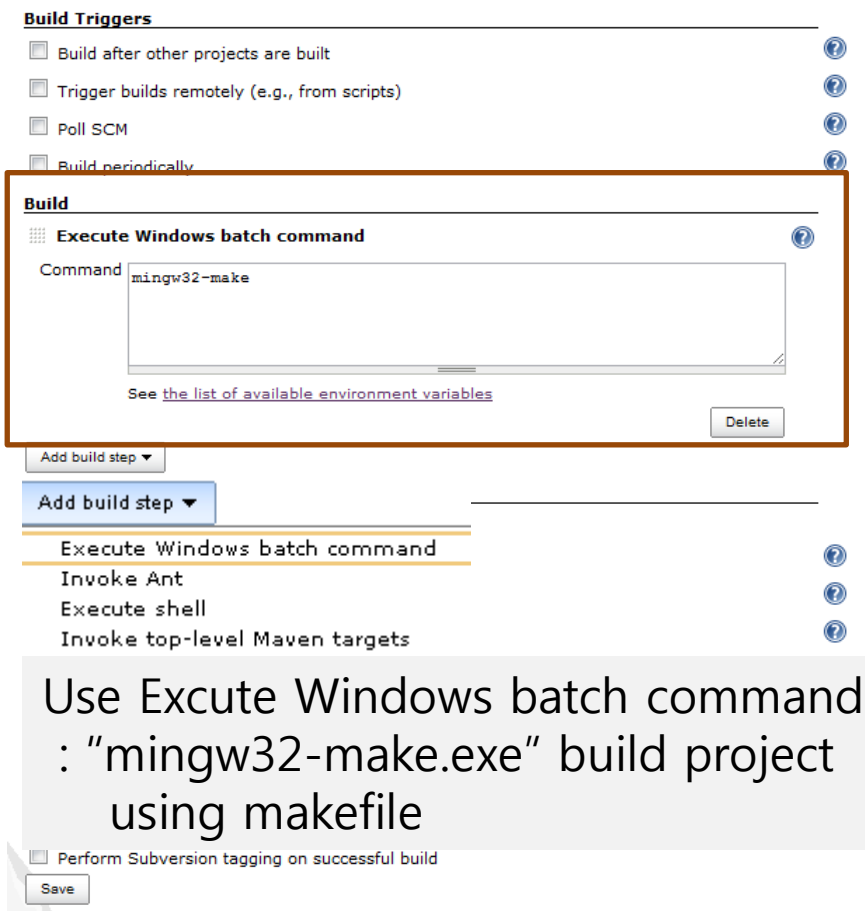


CI Server

❖ Job Configure



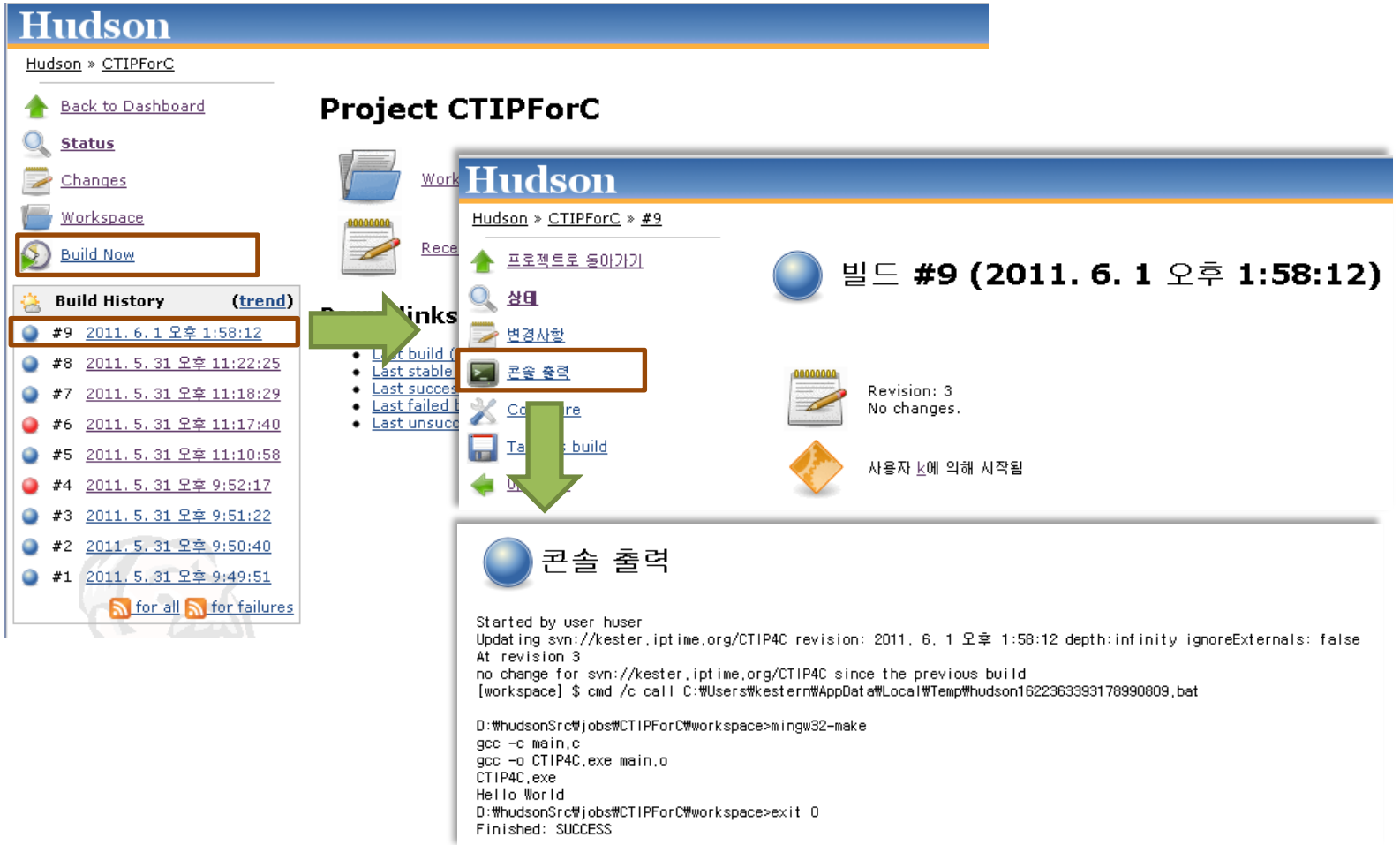
Enroll Repository



Use Excute Windows batch command
: "mingw32-make.exe" build project using makefile

CI Server

- Build



Unit Test - CUnit

CUnit

- Test Process
  1. Write functions for testcases
     - suite init/cleanup if necessary
  2. Initialize the test registry
     - CU_initialize_registry()
  3. Add suites to the test registry
     - CU_add_suite()
  4. Add tests to the suites
     - CU_add_test()
  5. Run tests using an appropriate interface
     - CU_basic_run_tests()
     - CU_automated_run_tests()
     - CU_console_run_tests()
  6. Cleanup the test registry
     - CU_cleanup_registry

CUnit

- Write functions for Testcases
  - Using CUnit Assertions
    - Similar with JUnit
    - Defined by CUnit.h

- Assertion Functions (1/2)

CU_ASSERT(int expression) **CU_TEST**(int expression)	Assert that *expression* is TRUE (non-zero)
CU_ASSERT_TRUE(value) **CU_ASSERT_FALSE**(value)	Assert that *value* is TRUE (non-zero) Assert that *value* is FALSE (zero)
CU_ASSERT_EQUAL(actual, expected) **CU_ASSERT_NOT_EQUAL**(actual, expected))	Assert that *actual* = = *expected* Assert that *actual* != *expected*
CU_PASS(message)	Register a passing assertion with the specified message. No logical test is performed.
CU_FAIL(message)	Register a failed assertion with the specified message. No logical test is performed.

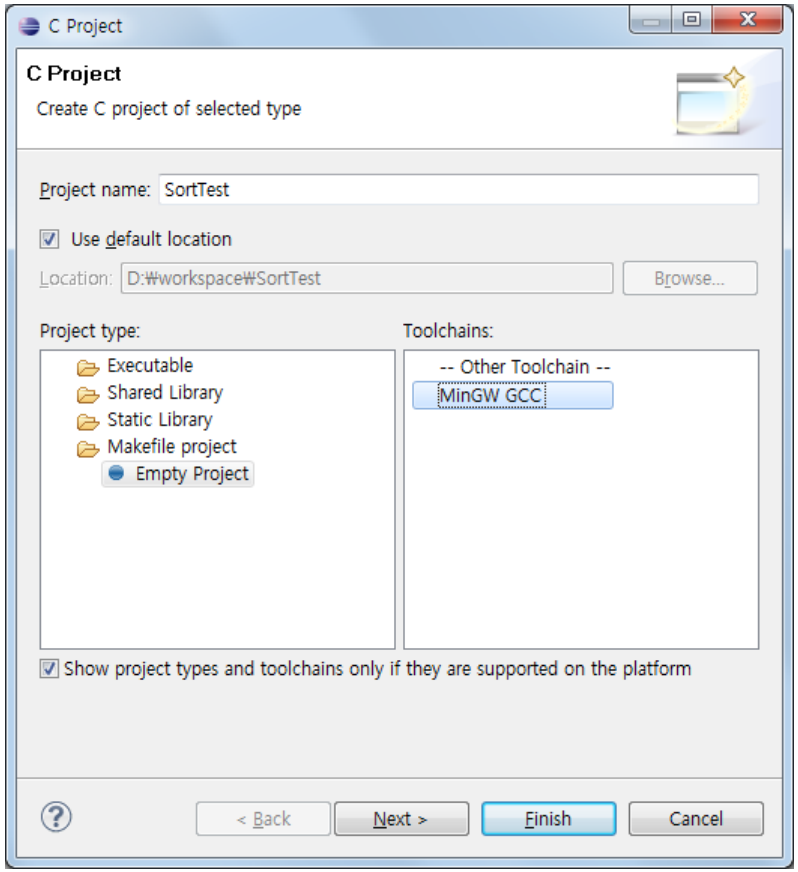
CUnit

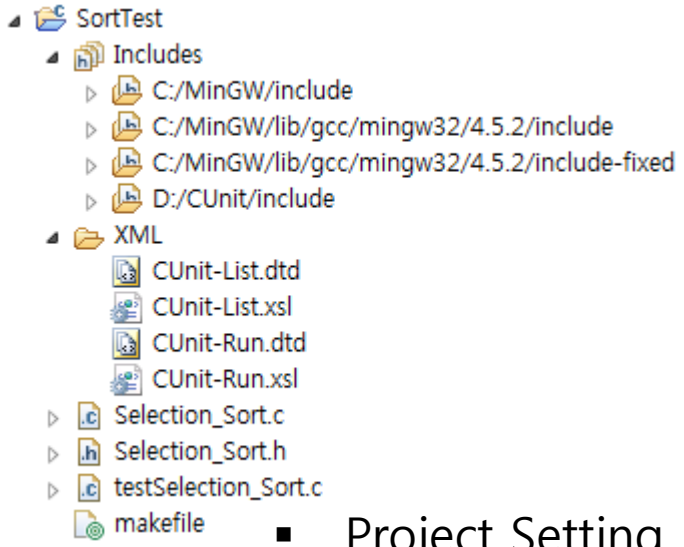
❖ Assertion Functions (2/2)

CU_ASSERT_PTR_EQUAL(actual, expected) CU_ASSERT_PTR_NOT_EQUAL (actual, expected)	Assert that pointers *actual* == *expected* Assert that pointers *actual* != *expected*
CU_ASSERT_PTR_NULL(value) CU_ASSERT_PTR_NOT_NULL(value)	Assert that pointer *value* == NULL Assert that pointer *value* != NULL
CU_ASSERT_STRING_EQUAL(actual, expected) CU_ASSERT_STRING_NOT_EQUAL (actual, expected)	Assert that strings *actual* and *expected* are equivalent Assert that strings *actual* and *expected* differ
CU_ASSERT_NSTRING_EQUAL (actual, expected, count) CU_ASSERT_NSTRING_NOT_EQUAL (actual, expected, count)	Assert that 1st count chars of *actual* and *expected* are the same Assert that 1st count chars of *actual* and *expected* differ
CU_ASSERT_DOUBLE_EQUAL (actual, expected, granularity) CU_ASSERT_DOUBLE_NOT_EQUAL (actual, expected, granularity)	Assert that |*actual* - *expected*| <= | *granularity*| Assert that |*actual* - *expected*| > | *granularity*| *Math library must be linked in for this assertion.*

Sort Test

❖ Create Project





- Project Setting
  - include directory
  - library path
- copy from CUnit/share
- add Files to Test
- add Testing file
- add makefile for Build

Sort Test

❖ Write Testcase

▪ Testsuite1

```
void test_exp(void) {
    char array[MAX_SIZE]="abedea";
    selection_sort(array);
    CU_ASSERT(array[0] == 'a');
    CU_ASSERT(array[1] == 'a');
    CU_ASSERT(array[2] == 'b');
    CU_ASSERT(array[3] == 'd');
    CU_ASSERT(array[4] == 'e');
    CU_ASSERT(array[5] == 'e');
}
void test_true(void) {
    int isEqual;
    char array[MAX_SIZE]="adebcf";
    selection_sort(array);
    if(strcmp(array,"abcdef")==0)
        isEqual = TRUE;
    else
        isEqual = FALSE;
    CU_ASSERT_TRUE(isEqual);
}
```

```
void test_equal(void){
    char array[MAX_SIZE]="abedea";
    selection_sort(array);
    CU_ASSERT_EQUAL(array[0],'a');
    CU_ASSERT_EQUAL(array[1],'a');
    CU_ASSERT_EQUAL(array[2],'b');
    CU_ASSERT_EQUAL(array[3],'d');
    CU_ASSERT_EQUAL(array[4],'e');
    CU_ASSERT_EQUAL(array[5],'e');
}
void test_string(void){
    char array[MAX_SIZE]="abedea";
    selection_sort(array);
    CU_ASSERT_STRING_EQUAL(array, "abcdef");
}
void test_fail(void){
    char array[MAX_SIZE]="abedea";
    selection_sort(array);
    CU_FAIL("fail test_fail");
}
```

▪ Testsuite2

```
void test_int(void) {
    char array[MAX_SIZE]="582547";
    selection_sort(array);
    CU_ASSERT_STRING_EQUAL(array, "245578");
}
void test_capital(void) {
    char array[MAX_SIZE]="EMCDOH";
    selection_sort(array);
    CU_ASSERT_STRING_EQUAL(array, "CDEHMO");
}
void test_special(void) {
    char array[MAX_SIZE]="(^@=#>";
    selection_sort(array);
    CU_ASSERT_STRING_EQUAL(array, "#(*=>@^");
}
void test_mix(void){
    char array[MAX_SIZE]="1a6V#e";
    selection_sort(array);
    CU_ASSERT_STRING_EQUAL(array,"#16Vae");
}
```

Sort Test

❖ Write main Function for Testing

```
int main()
{
    CU_pSuite testSuite=NULL;

    /* initialize the CUnit test registry */
    CU_initialize_registry();

    /* add a suite to the registry */
    testSuite = CU_add_suite("ASSERT_Test", NULL, NULL);

    /* add the tests to the suite */
    CU_add_test(testSuite, "ASSERT TEST", test_exp);
    CU_add_test(testSuite, "ASSERT TRUE TEST", test_true);
    CU_add_test(testSuite, "ASSERT EQUAL TEST", test_equal);
    CU_add_test(testSuite, "ASSERT STRING TEST", test_string);
    CU_add_test(testSuite, "ASSERT Fail Test", test_fail);

    testSuite = CU_add_suite("Type_Test", NULL, NULL);
    CU_add_test(testSuite, "TYPE INT TEST", test_true);
    CU_add_test(testSuite, "TYPE CAPITAL TEST", test_equal);
    CU_add_test(testSuite, "TYPE SPECIAL TEST", test_string);
    CU_add_test(testSuite, "TYPE MIX TESTt", test_fail);

    /* Run all tests using the automated interface */
    CU_automated_run_tests();
    CU_list_tests_to_file();

    /* Clean up registry and return */
    CU_cleanup_registry();
    return CU_get_error();
}
```

Declaration Suite Variable

Initialize Registry

add Suite to Registry

add Testcases to Suite

Run Test - XML Output

Clean up Registry

Sort Test

❖ Write makefile

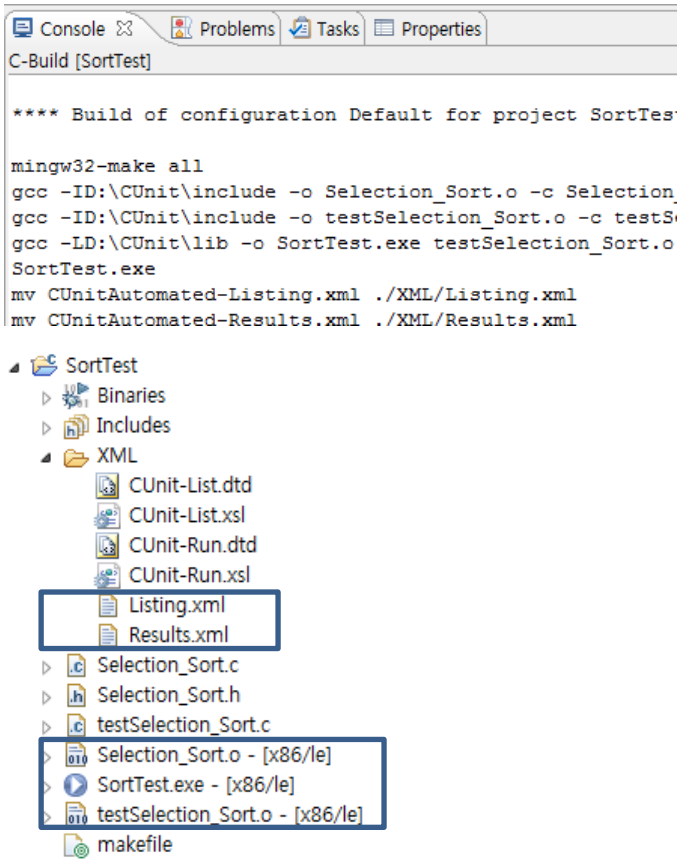
```
all : compile link run move
compile : Selection_Sort.c testSelection_Sort.c
    gcc -ID:\CUnit\include -o Selection_Sort.o -c Selection_Sort.c
    gcc -ID:\CUnit\include -o testSelection_Sort.o -c testSelection_Sort.c
link : testSelection_Sort.o Selection_Sort.o
    gcc -LD:\CUnit\lib -o SortTest.exe testSelection_Sort.o Selection_Sort.o -lcunit
run : SortTest.exe
    SortTest.exe
move : CUnitAutomated-Listing.xml CUnitAutomated-Results.xml
    mv CUnitAutomated-Listing.xml ./XML/Listing.xml
    mv CUnitAutomated-Results.xml ./XML/Results.xml
clean : Selection_Sort.o testSelection_Sort.o SortTest.exe
    rm Selection_Sort.o
    rm testSelection_Sort.o
    rm SortTest.exe
```

- compile : gcc with -I option
- link : gcc with -L option
- run : generate Test Result(XML)
- move : moving XML files
- clean : delete object, executable files

Sort Test

- Build makefile

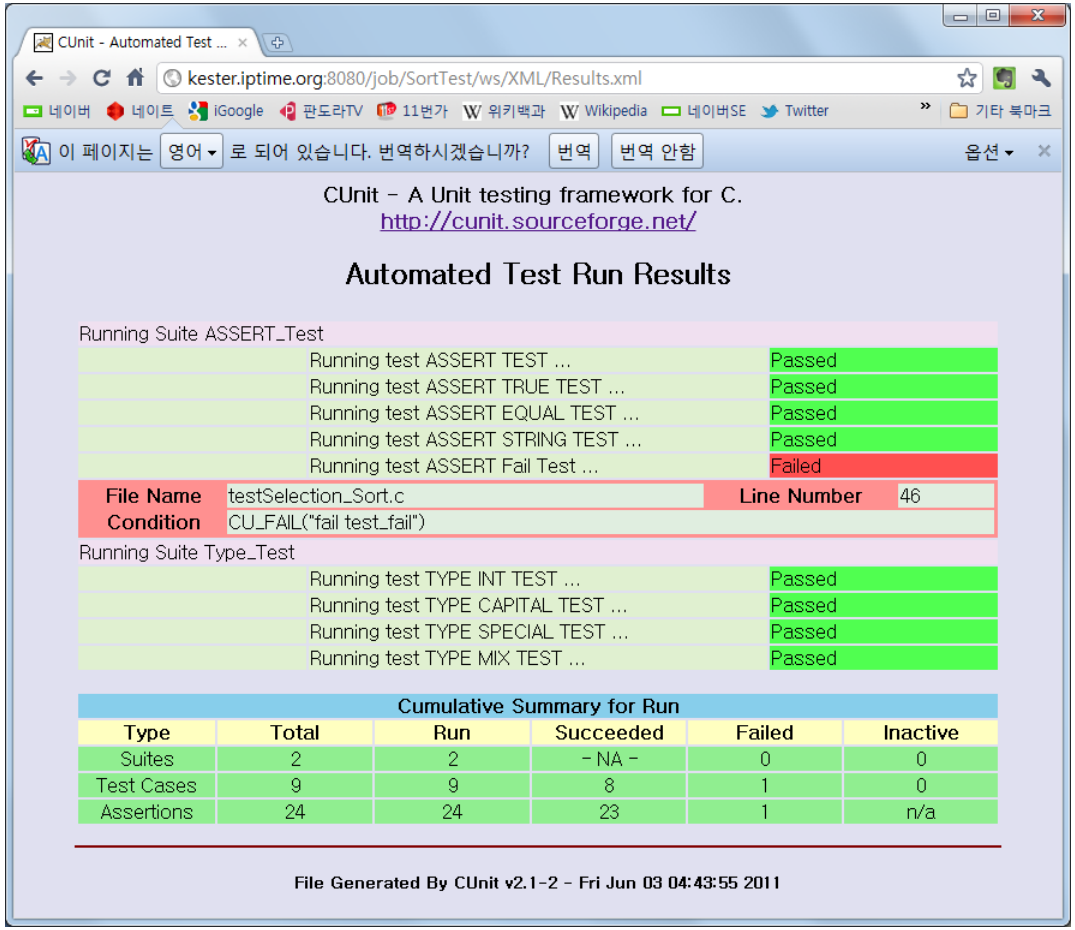
In Eclipse



In Hudson

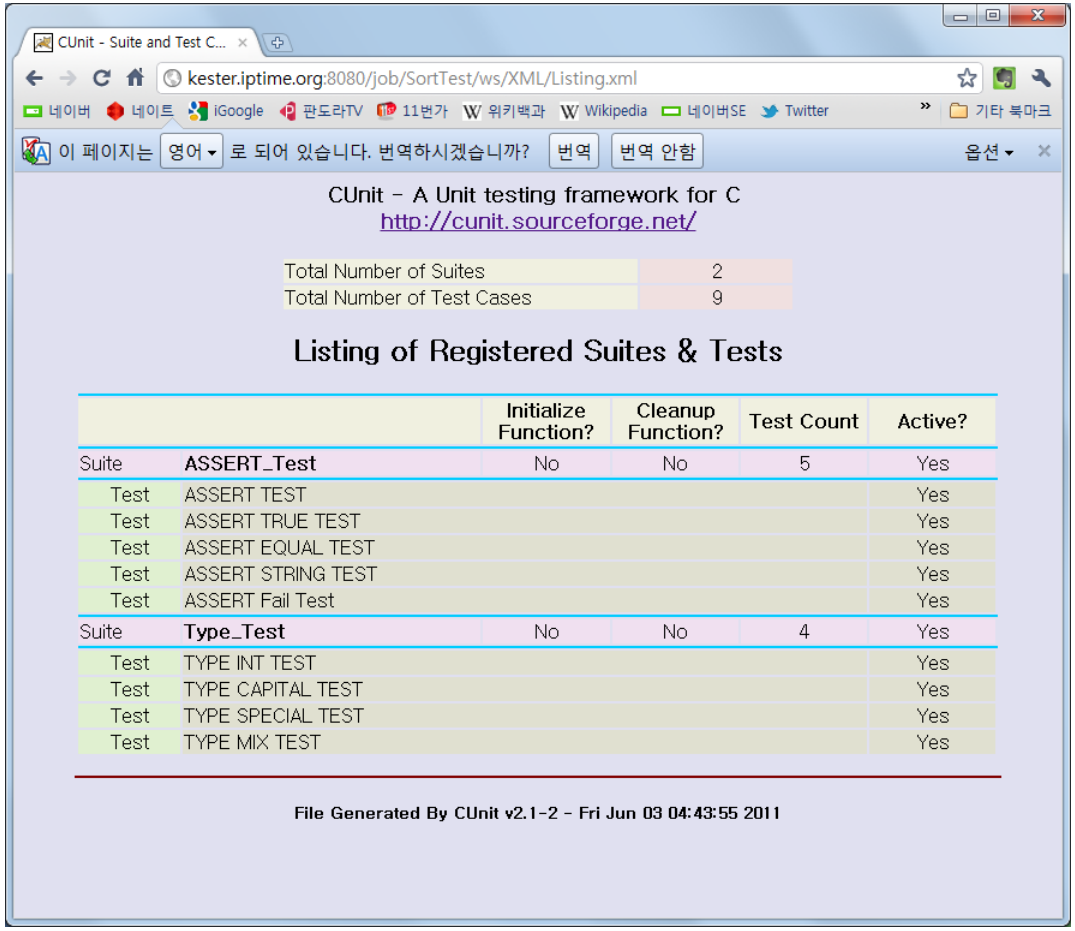
Sort Test

- Test Run Result



Sort Test

- List of Test



END