ECE 3574: Integration Testing

Changwoo Min

Let's work on Milestone 2

- Instructor's Office Hours
  - Mondays 10:30AM - Noon
  - Fridays 2-4PM

Recap Milestone 0 and 1

```
    .data
var:   .word 305419896
          # 0x12345678

    .text
main:
       lw $t0, var
end:
       j end
```

- Lexer (lexical analysis)
  - Tokenize an input string and produce a TokenList
- Parser (syntactic analysis)
  - Test if an input TokenList complies with the given BNF rules

Milestone 2

- **Develop a MIPS assembly simulator and its text-mode interface**
- **Three main components**
    - Lexer: the same as milestone 0
    - Parser: need to extend to produce information for VirtualMachine
        - initial memory for .data
        - program instructions for .text
        - labels for .data or .text
    - Virtual machine: our MIPS assembly simulator

Virtual Machine

1. MIPS registers (0-31, `pc`, `hi`, and `lo`)
2. Memory for .data (a sequence of bytes, 512 bytes)
3. Association between labels and memory locations or instructions
4. The program, as a sequence of insturctions

After parsing

Assembly

```
        .data
var:    .word  305419896
           # 0x12345678
var2:   .word 3405692606
           # 0xcafebebe

        .text
main:
        lw $t0, var
        lw $t1, var2
end:
        j end
```

Initial memory

Data addr	Byte
0	0x78
1	0x56
2	0x34
3	0x12
4	0xbe
5	0xbe
6	0xfe
7	0xca

Instructions

Code addr	Instruction
0	lw $t0, var
1	lw $t1, var2
2	j end

Labels

Label	Type	Addr
var	.word	0
var2	.word	4
main	.text	0
end	.text	2

Initial status of Virtual Machine

Data memory

Data addr	Byte
0	0x78
1	0x56
2	0x34
3	0x12
4	0xbe
5	0xbe
6	0xfe
7	0xca

Instructions

Code addr	Instruction
0	lw $t0, var
1	lw $t1, var2
2	j end

Labels

Label	Type	Addr
var	.word	0
var2	.word	4
main	.text	0
end	.text	2

Registers

Register	Value
$pc	0
$t0	0
$t1	...

Assembly

```
       .data
var:   .word  305419896
          # 0x12345678
var2:  .word 3405692606
          # 0xcafebebe

       .text
main:
       lw $t0, var
       lw $t1, var2
end:
       j end
```

Next step for design before coding

- Manually simulate test assembly files
  - `milestone2/tests/vm/*.asm`

Integration Testing

- Today we will take a look at integration testing and QtTest.
    - Techniques for testing command-line applications
    - GUI Testing using QtTest
    - Examples
    - Exercise

Recall our discussion of unit tests

- Unit tests exercise each module, generally a class and associated functions.
- Treat the public interface as a contract. Your test code checks the contract.

Integration Tests

- Integration tests verify the function of assemblies of modules or an application overall.

Functional testing of non-interactive applications

- Non-interactive applications which read files and write files specified through arguments are easy to test.
- You write another application to read the output and compare it to the expected output.
- For example consider a non-interactive application that reads input file and writes an output file taken as command-line arguments.

Functional testing of non-interactive applications

- In CMake

```
add_executable(the_app the_app.cpp)
add_exectuable(compare_tool compare_tool.cpp)
add_test(runtest1 the_app inputfile outputfile)
add_test(comparetest1 compare_tool outputfile
         ${CMAKE_SOURCE_DIR}/output1.expected)
```

- where the file outputfile.expected lives in the source directory.

Functional testing of interactive Text-Mode applications

- For simple interactive applications you can pipe in standard input and pipe out standard output.

```
$ my_exe < stdin_file > stdout_file
```

- For more complex interactive text-mode applications, e.g. a REPL, you can use a scripting language like Expect (Tcl) or Pyexpect (Python).

Testing using QtTest

- Tests are defined as the *private slots* of a class derived from QObject.
- A simple example using a single cpp file: mytest.cpp

```
class MyTest: public QObject
{
    Q_OBJECT

private slots:
    // define as many tests as you like
    void test1() { QVERIFY(true); };
};

QTEST_MAIN(MyTest)
#include "mytest.moc"
```

- This could be used for unit tests in the same way as Catch.

Assertions in QtTest are similar to those in other testing frameworks, e.g. Catch

```
QCOMPARE(actual, expected)
QVERIFY(condition)
QVERIFY2(condition, message)
QVERIFY_EXCEPTION_THROWN(expression, exceptiontype)
```

Testing a QtGui

- QTTest can be used for general testing but it really shines for Qt GUI testing because it can plug into the object tree and signal-slot mechanism.
- You can simulate Clicks and KeyPress events to get objects to handle events and emit signals as if they were triggered manually.
    - `QTest::keyClick()`
    - `QTest::keyPress(), QTest::keyRelease()`
    - `QTest::mouseClick()`, etc.
- See the Qt documentation for details.

To simulate events on a widget you need a pointer to it

- You can search for widgets using `QObject` (templated) find members

```
T findChild(const QString &name)
QList<T> findChildren(const QString &name)
QList<T> findChildren(const QRegularExpression &re)
```

- where the argument is the (optional) name property of the widget being searched for or a Perl-compatible regular expression for matching names.
  - `T` is the sub-type of `QObject`
  - by default this is done recursively

Example: find a pointer to a widget by type alone

- See `TestExampleWidget::testFindByType` in `test_example_widget.cpp`

Example: find a pointer to a widget by name alone

- See `TestExampleWidget::testFindByName` in `test_example_widget.cpp`

Example: find a pointer to some widgets by regular expression

- See `TestExampleWidget::testFindByTRegExp` in `test_example_widget.cpp`

Integration of CMake and QtTest

- Similar to configuring any Qt app from CMake

```
set(CMAKE_AUTOMOC ON)
set(CMAKE_INCLUDE_CURRENT_DIR ON)
find_package(Qt5 COMPONENTS Test REQUIRED)

add_executable(mytest mytest.cpp)
target_link_libraries(mytest Qt5::Test)

enable_testing()
add_test(mytest mytest)
```

- You run the tests manually or through cmake.

Exercise

- See website
- Qt Test Overview
- QTest Namespace
- QRadioButton
- QPushButton

Next Actions and Reminders

- Read about Design Patterns
- Enjoy your Spring Break (and Milestone 2)!