EECS 4314 Lab Notes

Part I: Source Code Download and Environment Configuration

1. Download the compressed file from: **www.cse.yorku.ca/~zmjiang/teaching/eecs4314/lab/eecs4314LabDemo.zip**
2. Make sure you have Java (at least 1.6 or above) installed
    - Open a command prompt and type "java -version"
3. Decompress the zip file to a directory (e.g., eecs4314/lab/)

Part II: Fact Extraction (This part is optional):

1. Uncompress the tar balls, use: **tar -xzvf c488.tar.gz** (for Linux/Mac/cgwin), some tools (e.g., WinRAR)

    C488 is a small compiler used in a compiler course at the University of Toronto. This program is small enough to be comprehensible, but big enough to make the reverse engineering of it interesting.
2. Open Understand and create a new project
3. Point understand to the directory which contains the c488 source code (~/eecs4314/lab/c488)
4. Understand fact extraction data export
    - **Reports -> Dependencies -> Export File Dependencies -> Export CSV**
    - Make sure you check the box "relative name"
    - Name the csv file to "c488_UnderstandFileDependency.csv"
    - Check the Appendix, if you are interested to understand the content
5. Download and install Perl
    - To verify the installation, open a command prompt and type "perl -v"
6. Run the following command in the command prompt. It should generate a file called **c488_UnderstandFileDependency.raw.ta.** Check [5] if you are interested to know more about the Tuple-Attribute (TA) language and format.
    - perl transformUnderstand.pl c488_UnderstandFileDependency.csv

Part II: Architecture Analysis and Visualization:

1. Create a hierarchical subsystem decomposition
    - Create a containment file called **c488_UnderstandFileDependency.contain** (see next page for an example)
    - Add a new line for each of the newly created subsystem in **c488_UnderstandFileDepndency.raw.ta** (after the first line):

```
$INSTANCE Parser.ss cSubSystem
$INSTANCE BackEnd.ss cSubSystem
$INSTANCE FrontEnd.ss cSubSystem
$INSTANCE c488 cSubSystem
```

2. Impose the containment on the extracted data:
    - Run this batch script: **createContainment.bat** (Windows) **or** **createContainment.sh** (Linux or Mac)
3. Visualize using LSEdit
    - Run this batch script: **runLSEdit.bat** (Windows) or **runLSEdit.sh** (Linux or Mac)

Sample c488_UnderstandFileDependency.contain file

```
      contain c488 FrontEnd.ss
      contain c488 BackEnd.ss
      contain c488 c488/main.c
      contain c488 c488/common.h
      contain c488 c488/scanner.h
      contain c488 c488/symbol.h
      contain c488 c488/globalvars.h
      contain c488 c488/tokenDef.h
      contain c488 c488/machine.h
      contain BackEnd.ss c488/semantics.c
      contain BackEnd.ss c488/semanticext.h
      contain BackEnd.ss c488/semantics.h
      contain BackEnd.ss c488/codegen.c
      contain BackEnd.ss c488/codegen.h
      contain BackEnd.ss c488/symbol.c
      contain BackEnd.ss c488/machine.c
      contain BackEnd.ss c488/machineDef.h
      contain FrontEnd.ss Parser.ss
      contain FrontEnd.ss c488/scanner.c
      contain Parser.ss c488/parser.c
      contain Parser.ss c488/parser.h
      contain Parser.ss c488/parsecodegen.h
      contain Parser.ss c488/parsetables.h
      contain Parser.ss c488/scanparse.h
      contain Parser.ss c488/parsesemantics.h
      contain Parser.ss c488/tset.c
      contain Parser.ss c488/tset.h
```

EECS 4314 Lab Notes
LSEdit Demo

1. Basics: File Menu, Navigation, Navigation Aids
2. Operations:
    a) Resize, Drag
    b) (Under Edit) Undo and Redo
    c) (Under Explore) Forward/Backward query
        - Forward query: To find out what subsystems current subsystem depends on
        - Backward query: To find out what subsystems require the current subsystem
    d) Select/unselect a box, mouse over, Right/Left click
    e) Expand the subsystem (e.g., backend.ss)
        - You can either example the “little square icon” on the top left, or double-click the box
        - Once expanded, you can also perform forward/backward queries
    f) If you are currently inside a subsystem (e.g., backend), you can always go up one level by clicking the upper left icon
    g) If you have too many subsystems (e.g., within backend), you can also try the automated layout and clustering options under the Layout menu
    h) Graphical options (e.g., Fonts, arrows). There are many options to configure the look and feel of the architecture. For example, you can right click the box “c488/TokenDef.h” and explore the draw options.
    i) TOC, History, Map, Legend
    j) Arrows reflects weights, counts, lift edges
    k) More …

EEDS 4314 Lab Notes

References:

1. SciTools - Understand Static Analysis Tool. https://scitools.com
2. Grok/jGrok Documentation: http://swag.uwaterloo.ca/~nsynytskyy/grokdoc/index.html
3. Newest version of LSEdit and its documentation: http://www.swag.uwaterloo.ca/lsedit/index.html
4. Richard C. Holt. An Introduction to TA: the Tuple-Attribute Language, technical Report, University of Waterloo, 1997. http://plg.uwaterloo.ca/~holt/papers/ta-intro.doc
5. Dependency Graphs video from SciTools: https://scitools.com/support/dependency-graphs-video/
6. Nikita Synytskyy, Richard C. Holt, Ian J. Davis: Browsing Software Architectures With LSEdit. The 13th International Workshop on Program Comprehension. 2005.

EECS 4314 Lab Notes
Appendix: How to understand the “Understand” file-level dependency file

The exported dependency file follows the csv (comma-separated-value) file format. It contains data like the following:

```
From File,To File,References,From Entities,To Entities
c488/main.c,c488/parser.c,2,1,2
c488/parser.c,c488/scanner.c,2,2,1
…
```

The second line indicates that there is one function, called "main(int argc , char *argv[])" in main.c calls the following two functions in parser.c

- parserInitialize();
- parser();

Such information can be checked in two approaches:

- by looking into the two source code files, or
- right-click the file name "main.c" -> "View Dependencies". The dependency browser should show in the lower right part of Understand

The third line indicates that there are two different functions (parser() and parserInitialize()) in parser.c, which reference the same function (scannerAdvance()) in scanner.c.