The Mimetic Methods Toolkit

A self-included, hands-on introduction

http://csrc.sdsu.edu/mtk/

Eduardo J. Sánchez P.

Ph.D. Student / Teaching Associate
Computational Science Research Center (CSRC) at SDSU

COMP-693-B: Advanced Num. Analysis, Guest Lecture 1
San Diego State University’s College of Sciences
April 15, 2013, San Diego, California
Part 0: Prelude in C... ++

1. ACM – E.1.1.3 – Operating systems:
   – An introduction to UNIX-like operating systems

2. ACM – E.3.3.2 – Object oriented development:
   – An introduction to Object oriented development

3. ACM – E.2.1.1.4 – Object oriented languages:
   – An introduction to C++

4. ACM – E.2.3 – Compilers:
   – Program building in UNIX and the make tool
AN INTRODUCTION TO UNIX-LIKE OPERATING SYSTEMS

Reference: [1]
UNIX and shells

• The first version of UNIX came from AT&T in the early 1970s
• UNIX was developed by programmers and for programmers
• UNIX is designed so that users can extend its functionality
  – To build new tools easily and efficiently
  – To customize the shell and user interface
  – To string together a series of UNIX commands to create new functionality
  – To create custom commands that do exactly what is needed
Evolution of UNIX and UNIX-like systems

UNIX architecture and hierarchical layers

Source: http://torrentfreaknews.blogspot.com/2013/01/what-is-unix-shell.html
UNIX and shells

• A shell is:
  – A command interpreter that turns text that you type (at the **command line**) into actions

• Shells can:
  – Customize a UNIX session
  – Allow for scripting
UNIX and shells

• Two main flavors of UNIX shells:
  – Bourne: sh, ksh, bash, zsh
    • AT&T shell
    • Fast
    • Use ‘$’ for command prompt
  – C shell: csh, tcsh
    • Berkley Shell
    • Better for user customization and scripting

• To check shell:
  $ echo $(SHELL)

• To switch shell:
  $ exec bash
The Shell Landscape

Source: http://www.novell.com/coolsolutions/feature/11232.html
UNIX commands

• Commands:
  – **ls** – Lists the content of a directory
    
    ```
    $ ls
    ```
  – **mkdir** – Makes a directory
    
    ```
    $ mkdir my_directory
    ```
  – **cd** – Changes directory
    
    ```
    $ cd my_directory
    ```
  – **cp** – Copies into
    
    ```
    $ cp ./my_directory/file.txt ../file_in_parent_folder.txt
    $ cp ./my_directory/file.txt ./same_file_here.txt
    ```
  – **rm** – Removes
    
    ```
    $ rm ../file_in_parent_folder.txt
    ```
UNIX commands

• Notice that:
  – ‘.’ means “current working directory”
  – “..” means “parent working directory”
  – ‘~’ means “home directory”

• Two more commands:
  – pwd – Path to working directory
    $ pwd
  – man – Manual
    $ man ls
UNIX files

• A file is the basic component for data storage  
  – UNIX considers everything it interacts with a file

• A file system is UNIX’s way of organizing files on mass storage (disk) devices  
  – A physical file system is a section of the hard disk that has been formatted to hold files

• The file system is organized in a hierarchical structure similar to an “inverted” tree
Typical UNIX Directory Structure

Source: http://cnx.org/content/m13327/latest/
ANY QUESTIONS SO FAR?
Suggested practice exercises

EXERCISE 0X01
Exercise 0x01

• Create the following file structure in your home folder:
  1. home/
     1. your_user_name/
        1. libraries/
           1. example_api/
              1. examples/
              2. include/
           2. include/
            3. lib/
        4. src/
        2. another_example_api/

• Can you discuss its semantic?
AN INTRODUCTION TO OBJECT ORIENTED DEVELOPMENT

Reference: [2]
Object oriented development

- In **Object oriented development (OOD)** programs are conceptualized as interacting **objects**
  - Objects contain **attributes** to describe their state
  - Objects contain **methods** to describe their behavior
  - Program design phase involves designing objects and their attributes and methods
Object oriented development

• Encapsulation
  – Objects contain their own attributes and methods
  – Information hiding

• Polymorphism
  – A single name can have multiple meanings depending on its context and namespace

• Inheritance
  – Objects can inherit characteristics from other objects
  – Writing reusable code
AN INTRODUCTION TO C++
C++

- C was developed by Dennis Ritchie at AT&T Bell Labs in the 1970s
  - Used to maintain UNIX systems
  - Many commercial applications written in C
- C++ was developed by Bjarne Stroustrup at AT&T Bell Labs in the 1980s
  - Overcame several shortcomings of C
  - Incorporated Object oriented development
  - C remains a subset of C++
C++

C = C + 1;
C++;
C++

• A simple C++ program begins this way:

```cpp
#include <iostream>

using namespace std;

int main() {
    cout << "Hello, world!" << endl;
    return 0;
}
```
C++

• A simple C++ program begins this way:

```cpp
#include <iostream>

using namespace std;

int main() {
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```
C++

• Include Directives

```
#include <iostream>
```

– Tells compiler where to find information about items used in the program

– iostream is a **library** containing definitions of cin and cout

• Variable declaration line example:

```
int number_of_pods, peas_per_pod, total_peas;
```
Reference: [2]

PROGRAM BUILDING IN UNIX AND THE MAKE TOOL
Building programs

• **Build a program:**
  • **Compilation:**
    – Syntactic analysis
    – Converting the general programming language to an architecture-specific assembly code and then to machine code
  • **Linking:**
    – Grabbing all of the chunks of machine code and putting them together to create the program
A hierarchical relationship among languages

BOOL CMycf29BAuto::DisplayDialog()
{
    // TODO: Add your dispatch handler code here
    TRACE("Entering CMycf29BAuto::DisplayDialog \n\n", this);
    BOOL bRet = TRUE;
   AtlUnlockTempMaps();  // See MFC Tech Note #3
    CWnd* pTopWnd = CWnd::FromHandle(::GetTopWindow(NULL));
    try
    {
        CPromptDlg dlg;  // (pTopWnd)
        if (n_vaTextData.wt == VT_BSTR)
        {
            // converts double-byte character to single-byte character
            dialog.m_strData = n_vaTextData.bstrVal;
        }
        dialog.nIData = n_nData;
        if (dlg.mModal() == IDOK)
        {
            n_vaTextData = COleVariant(dialog.m_strData).Detach();
            n_nData = dialog.m_nData;
            bRet = TRUE;
        }
        else
        {
            bRet = FALSE;
        }
    }
    catch (CException* pe)
    {
        TRACE("Exception: failure to display dialog\n");
        bRet = FALSE;
        pe->Delete();
    }
    AtlUnlockTempMaps();
    return bRet;
}
2\textsuperscript{nd} Level: Architecture-DEPENDENT (MNEMONIC)

Source: http://www.wired.com/threatlevel/2012/03/duqu-mystery-language-solved/
MIPS32 Add Immediate Instruction

Equivalent mnemonic: \texttt{addi $r1, $r2, 350}

3\textsuperscript{rd} Level: Architecture-DEPENDENT (REAL)

The make tool

• Programs are built using:
  – make
  – We create Makefiles, which keep track of the changes in different modules of the program
  – Efficient building, since:
    • Changes are tracked
    • Only compile modified modules
    • We link at the end
ANY QUESTIONS SO FAR?
Suggested practice exercises

EXERCISE 0X02
Exercise 0x02

• Chapter 5, Section 6, Exercises 1 and 2 of:
Exercise 0x02.1

1. Implement a structured program, using C as the programming language, and which implements a Point3D data type:
   a) Specify and justify the attributes and the operations
   b) Write a client code that creates and orders a finite collection of randomly specified 3-D points based on their norm
   c) Normalize them all and print all of their norms to visually corroborate that the normalization has been successful
Exercise 0x02.2

2. Extend the previous problem to C++
   a) Specify and justify the classes, their attributes, and the operations
   b) Write a client code that performs the same operations as in Problem 1
   c) Justify your design by means of a UML class diagram…
      … some reading is required to learn UML! LOL!
      Have fun and happy coding!
References for Part 0

By Mark G. Sobell

By José E. Castillo, Guillermo F. Miranda

Eduardo Sánchez - Computational Science Research Center at SDSU, 2013
I'll believe in what the wind brings to me,
in pure love and great emotion... I will believe

- Luca Turilli et al, Triumph or Agony, 2006

THANKS! QUESTIONS?