Lecture 4
Getting Started with ITK!

Methods in Medical Image Analysis - Spring 2012
BioE 2630 (Pitt) : 16-725 (CMU RI)
18-791 (CMU ECE) : 42-735 (CMU BME)
Dr. John Galeotti

Based in part on Shelton's slides from 2006

Goals for this lecture

- Compile, compile, compile
  - Learn how to use SVN & CMake
  - Build ITK
  - Compile several programs that use ITK
- Find documentation online
- Learn the quirks (if any) of the system you choose to use

Getting help

- Email your TA
  - Do this sooner rather than later!
  - Vikas R.S.
  - vir16+miia@pitt.edu
- Email your instructor
- Join the insight-users mailing list; instructions are at http://www.itk.org

Assignments

- Collaboration is **encouraged**; unless told otherwise, feel free to discuss assignments with other students
- But… *please* submit your own code - don't copy and paste stuff from friends
- More so than other classes, you will be learning techniques that translate directly to the real world - don't cheat yourself

4

Grading of assignments

- Grading criteria:
  - Does it accomplish the specified task?
  - Is it well commented? Follow the "6 month rule" - if you leave for 6 months, you should be able to pick up where you left off.
  - Many/most assignments will be divided into sections, with each section pass-fail.
    - We may give opportunities to fix "stupid" problems before final judgment is passed

5

Assignments, cont.

- Please interpret due dates as absolute, unless told otherwise
- Really
- We're happy to spend time helping you debug code, but not at 11 pm the day before the assignment is due

6

Computer requirements: recommended

- Your own computer is preferable
  - Cluster machines should also work
  - Please be aware that ITK can consume a lot of disk space during the build process
- Windows, Visual Studio 2010, Python 2.7
  - We aren't trying to force everyone to use this, but…
  - This is what the grader will be primarily using.
  - On a Mac? Consider Parallels or VMware Fusion
  - Run Linux? Consider VMware Workstation

7

Alternative usable computer configurations

- Any platform supported by ITK (Mac, Linux, etc.)
- **If there are problems, you will have to work with the grader to get your code working on their machine.**
  - Try having the TA or grader check your code before it is due.
- If the grader's computer can't run your code, you will have a short (but reasonable) period of time to fix it after he emails you that your code appears broken (along with what errors he got).
  - If you are trying to make things work, but have many things to "fix," then more time may be granted.
  - For final projects, we *may* decide to let you show the TA your code running on your own machine, on a case-by-case basis.

8

What is ITK?

- To clarify, ITK is a *toolkit*
  - It doesn't "do" anything
  - You can't "run" it
  - There isn't an itk.exe file
- Typically, you use ITK in conjunction with other toolkits to handle visualization and GUI interaction

9

So, what's it good for?

- ITK code is easy to add to existing C++ code
  - Also Python, Java, …
- It provides a variety of flexible data containers, and ways of processing / analyzing them
- You can do a lot in only a few lines of code
- Once you get used to it, it's easy to use (gasp!)

10

What we assume you can do

- Understand C++ and/or Python syntax
  - Standard flow control such as for, do, calling functions, etc.
  - Classes
  - Inheritance
  - For C++: Pointers, dereferencing, passing by reference
- Work comfortably in the operating system of your choice, using the compiler or Python environment of your choice

11

You may have not…

- Used revision control using SVN (or CVS)
- Engaged in collaborative programming
- Written C++ code that builds on multiple platforms
- Used cross-platform make software
  - (CMake or Jam, for example)
- Designed software using a data-flow architecture, worried about smart pointers, etc.

12

Revision control with SVN

- Revision control software allows you to store incremental changes to software
- You will be expected to use SVN to manage your homework assignments
  - SVN is like CVS, but better
- I encourage you to use revision control on your code outside of this class as well - it's a good habit to develop

13

SVN terms

- Server - what it sounds like
- Module - a group of files that can be accessed on the server
- User - each module has associated users, with varying levels of access (read only, read/write, etc.).

14

SVN terms, cont.

- Checkout - Download a fresh copy of a module from the server to your computer
- Update - Sync your copy of a module with the server copy; much faster than a checkout
- Commit - Merge changes made to your local copy with the server

15

SVN setup

- The SVN server for this course will be:
  - https://svn.vialab.org/svn/
- You will each have a module, based on your email; you will get email about this in a week or two.
- Only you and the instructors will have access to this module

16

SVN setup, cont.

- GUI wrappers for SVN
  - Windows: Tortoise SVN
    - http://tortoisesvn.net/
  - Mac: svnX
    - http://www.apple.com/downloads/macosx/development_tools/svnx.html
  - Windows, Mac, Linux, etc: RapidSVN
    - http://rapidsvn.org/download/release/0.12/
- Command line works fine too, but may be more awkward if you're used to GUI's

17

Cross platform (C++) development

- ITK builds on a large combination of operating systems and platforms
- For C++, each compiler has it's own input format: Makefiles, workspaces, etc.
- Q: How can you possibly coordinate builds on different platforms?

18

The answer: CMake

- Cross platform tool to manage the build process
- Simplifies the build process
- Auto-configuration
- Easy access to external libraries
- Used by several other open source projects

www.cmake.org

CMake is:

- Required to build native (C++) ITK
- Cross-platform project generator
- Often simpler than particular environments
- Text as input
- Project file as output:

Windows	Visual Studio Solution
UNIX	Makefile
Mac OS X	Xcode project or Makefile

How CMake runs

- Write a CMakeLists.txt file describing your project in CMake's language
- Run CMake to generate an appropriate makefile/project/workspace for your compiler
- Compile as you normally would

How CMake runs, cont.

- This is not unlike the configure-make process you may be familiar with from various Unix systems
- But… it works with many compilers
- CMakeLists.txt files are easy to perform revision control on

22

CMakeLists.txt syntax

- Comment lines indicated with #
- Look at examples in ITK
- Simple example:

```
cmake_minimum_required(VERSION 2.4)
#Make sure the user's CMake is recent enough

#Give this project a name:
PROJECT(cool_demo)

#The command-line executable "demo"
#is built from "demo_code.cxx" and
#must be linked with the ITK libraries
ADD_EXECUTABLE(demo demo_code.cxx)
TARGET_LINK_LIBRARIES(demo ${ITK_LIBRARIES})
```

23

Steps to get started with ITK

- Pay Attention
- This is part of HW2

24

Step 0 - Don't panic!

- There is *substantial* documentation on everything I'm going to present here, and vastly more about things that we will never cover in this course
- http://www.itk.org/ITK/help/documentation.html
- Download a copy of the *ITK Software Guide*

25

Step 1 - Install CMake

- Check if a recent version of CMake is already installed on your computer.
- If not, …
- Download and install a binary distribution of CMake 2.8.7 from:
  - http://www.cmake.org/

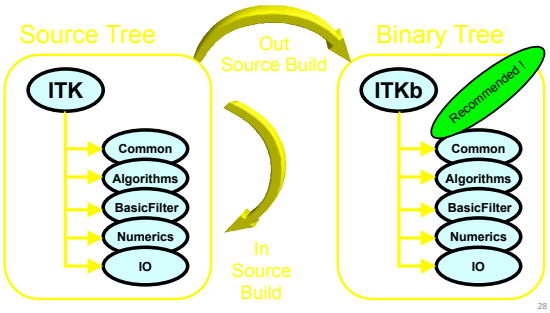
26

Step 2 - Install ITK

- Check if ITK 4.0 is already installed on your computer.
- If not, …
- Download the latest version of InsightToolkit:
  - http://www.itk.org/ITK/resources/software.html
- Extract, e.g., InsightToolkit-4.0.0.zip to your working source directory for this class

27

In source vs. out source builds



28

Why use two trees?

- Keeps your C++ source and binary code separate
- Minimizes the amount of damage you can do to your SVN tree
- ITK is found in the InsightToolkit-4.0.0 folder
- We suggest that you build it in a new folder you create named InsightBin

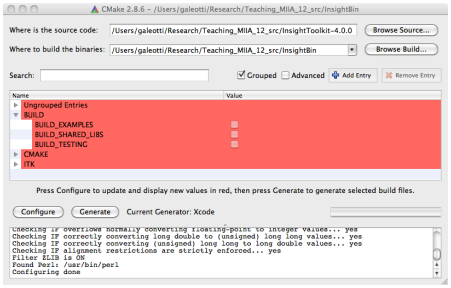
29

Configure - Easy Start

- Run CMake
- Select the SOURCE directory
- Select the BINARY directory

30

Configure - Easy Start, cont.



31

Configure - Easy Start, cont.

- Disable BUILD_EXAMPLES
- Disable BUILD_TESTS
- Disable BUILD_SHARED_LIBS

32

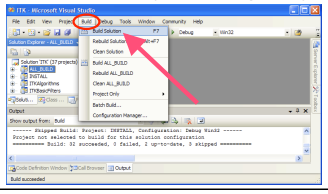
Configuring and Generating

- Each time you change an option or options you may need to "configure" CMake again
- If the generate option ("OK" under Windows) is not presented, you definitely need to hit configure again
- If any of the options are highlighted in red, you need to reconfigure

33

Build ITK

- Open the ITK Visual Studio Solution file in the Binary Directory
- Select Build→Build Solution
- It will probably take somewhere between 10 - 40 minutes, but your mileage may vary



34

Verify the Build

Libraries will be found in:

ITK_BINARY / bin / { Debug, Release}

35

Building with gcc

- Order of operations is the same
- Differences
  - Run the ccmake executable, which uses a curses TUI, the options are identical
  - Run make instead of Visual Studio
  - Think of CMake as replacing the "./configure" step you may be used to

36

Building with gcc cont.

Start in directory *containing* InsightToolkit-4.0.0

```
mkdir InsightBin
cd InsightBin
ccmake ../InsightToolkit-4.0.0
```

Edit CMake options

Reconfigure if needed

```
make
```

37

Now what?

- At this point, you should have two things:
  - A directory containing a bunch of source code
    - E.g. …/MIIA/InsightToolkit-4.0.0/
  - A directory containing the built ITK libraries
    - E.g. …/MIIA/InsightBin
- As mentioned earlier, you don't have anything executable

38

Building an application

- ITK comes with a simple application you can build in order to test the ITK libraries "out of source" (i.e. not built inside ITK)
- It can be found in:

  InsightToolkit-4.0.0/Examples/Installation

39

How to build HelloWorld

- Copy & rename the *Installation* directory somewhere outside of the Insight directory
- Run CMake on HelloWorld
  - Remember the source/binary distinction and use HelloWorldBin as your build location
- CMake should automatically find ITK
  - if not, edit the ITK_DIR option

40

How to build HelloWorld, cont.

- Once CMake is happy, generate the makefile/ project for your compiler
- Build HelloWorld
- Give it a try

41

More examples

- You can turn on ITK's *Examples* option in CMake, which will build all of the examples for you
- Or… you can copy the examples out-of-source and build them like you did HelloWorld
- These examples link into ITK Software Guide; read the chapter, poke the code and see what happens…

42

C++ Workflow thoughts

You should get used to the idea of:

1. Writing some code
2. Writing a CMakeLists.txt file
3. Running CMake
4. Building your code
5. Rinse, repeat

43

An aside: how to use ITK with existing C/C++ applications

- Your existing app may not use CMake
- In this case, you need to link to the ITK libraries explicitly and include the appropriate source directories
- This isn't hard, but it may take some trial and error to discover everything you need
- You don't need to worry about this in the context of this class

44

ITK Documentation

- Most of the ITK documentation is generated automatically from source comments using Doxygen
- Please familiarize yourself with the various means of navigating the Doxygen documentation online, e.g. "Alphabetical List":
- http://www.itk.org/Doxygen40/html/index.html

45