

Intro, not a full c++ course, just try and touch all the important bases.
You are welcome/encouraged to ask questions, this works best if I know what don't understand/know.

Different file types, .cpp .cxx .C .c .h .hh all plain text, just for programmers to know what's what

code0.cpp - Basics of a program

-Include libraries -namespace std -int main (definition of a function) -cout

TOGETHER: write a hello world, compile and execute

code1.cpp - input arguments

TOGETHER: write a for loop, test inputs

code2.cpp Definition of a function continued (and stringstream)

TOGETHER: Fix the function and test

code3.cpp

pointers carry only a type and location in memory of what they "point to"

Why is this useful?

When the object is large we do not want to copy it so we pass around a pointer to it is much much faster.

TOGETHER: Run original, look at output memory address

TOGETHER: Create an pointer, output, its address, its value and dereference

code4.cpp

Example of a crashing because we pointed to somewhere that doesn't exist because variables only exist in local memory (may not actually crash but warn it would in future.

"new" introduced creating things on the "heap"

Have to delete, but also dangerous as program trusts us there is something at the other end of the pointer.

Deleting something that isn't there, crash, some other part of program accessing a object that was deleted, crash

TOGETHER: Run, crash, read

TOGETHER: define "new" run and it is fixed

code5.cpp - show the heap addresses vs local addresses

TOGETHER: Run and look at output

(optional) INSTRUCTOR: Demonstrate variable reuse in first loop. Dont assume zero initiation!

code6.cpp -Basics of a class

constructor and destructor, methods and attributes, public and private

TOGETHER: Extend methods. What else might we like?

(Suggestions improve constructor, +, mag, theta)

TOGETHER: Use methods with ":" and cout

INSTRUCTOR: Show pointer method access by -> rather than dereference. Important!

code7.cpp - Inheritance

Behaves as parent, has all attributes and functions, with new things layered on top

parent constructor must be called explicitly to pass inputs

can have multiple inheritance

TOGETHER: Do ThreeVector operations using a mix of the three

Any questions. Totally skipped some topics. Really only a bare bones introduction.

Good time for a loo break.

ROOOOOOOT

Show root website examples and class guide:

<https://root.cern.ch/doc/master/classTH1.html>

<https://root.cern.ch/howtos>

<https://root.cern.ch/root/html/tutorials/graphs/zones.C.html>

<https://root-forum.cern.ch/t/how-to-delete-histogram/19834>

TOGETHER: Start a root terminal

The root terminal is a powerful interpreter which can run on the fly c++, load libraries and functions and run GUI things.

INSTRUCTOR: -demonstrate a loop typed out on command line

-Introduce a TCanvas, create a new one.

-Demonstrate autocomplete

-Demonstrate functions SetWindowSize and SetWindowPosition

Introduce TH1 class Types D F etc

TOGETHER: *Create a local TH1*

"I usually set variable name and "NAME" to be the same but I am not for this example"

Many classes inherit from the *TNamed* class meaning they should have a unique name.

The name is then associated with a pointer in the root interpreter environment **and** the name can be found by found by other root functions if we need to fetch the object.

INSTRUCTOR: *Demonstrate direct and pointer access*

TOGETHER: *Draw()* (can have options but we aren't using any just yet)

-Automatically creates a canvas as needed

-By default the canvas does not "own" the histogram, it is pointing to the histogram which is "owned" by whatever environment it was created by (in this case the main root environment), so when canvas is deleted (closed) the histogram is not effected.

INSTRUCTOR: *Close Canvas and Draw h0 again*

TOGETHER: *We can update histogram on the fly introduce histogram "Fill" command*

Ah but that didnt update until we clicked on it, because the canvas didn't know anything had been modified.

-Introduce gPad (points to whatever the currently selected canvas is. "selected" means the backend drawing system things it is currently aiming at for any subsequent drawing commands)

-When ever a new canvas is created, even automatically, gPad switches to pointing at that Canvas

TOGETHER: *gPad->Modified(). Do another Fill to check*

-gPad can be changed with a cd command or by clicking (the later being temperamental)

INSTRUCTOR: Demonstrate with a pair of canvases

TOGETHER: FillRandom("gaus",1000);

Now introduce the drawcopy command

INSTRUCTOR: show drawcopy command. Original no longer changes what is drawn

This is useful if we want see a copy of histogram midway through a filling operation or some other modification.

Note we do have pointer returned, so show we can still modify using pointer.

INSTRUCTOR: Show SetLineColor, GetXAxis

However with drawcopy the canvas owns the histogram. Caution.

INSTRUCTOR : show segmentation violation after canvas is deleted

Now introduce scripts

- "scripts" usually denoted .C are just a list of command line commands.

-Very useful for not losing your progress every time you segfault the interpreter

TOGETHER: Create one from script0.C (Just one that creates, FillRandom and Draws)

-can be executed from command line with .x

Now using GUI or command line make histogram pretty / Q&A / loo break

STUDENTS: Explore histogram formating tools

Loading functions and codes

Can load full libraries and functions using at start or with .X .L

.x loads (& interprets /soft compiles) then runs

.x will always look to execute a function with the name of the file following loading

INSTRUCTOR : .L on code2(b).cpp

TOGETHER: .L use threevec.h

See now our threevec class is usable by command line .See how autocomplete works too!

Can be used by any subsequent codes/functions loaded

TOGETHER: .x or .L func0.cpp

TFiles

Within the root environment you are always “in” some directory with respect to memory and ownership (more details following)

This is important because anything on the heap, which is roots default, will be deleted when its owner goes out of scope i.e. delete

e.g. a file is closed or is a local object inside a function that ends.

INSTRUCTOR: Show explicitly opening a file by creating a TFile

note: even file name is tab completable

READ, CREATE, RECREATE, UPDATE

INSTRUCTOR: show .pwd and .ls

note: when you open a file you automatically move in to it just like Canvases

TOGETHER: from command line _file0 .pwd .ls

TOGETHER: gROOT->cd() .pwd

cd (also same behavior structure as Canvases)

can do ls function of directories you arent in, .ls just automatically does it where you are

TOGETHER: _file0->ls()

So why does where you “are” in the environment matter?

Files being read:

TOGETHER: navigate into file _file0->cd()

When “in” a directory we can directly access objects by their “identifier/namecycle” like a pointer withing the root interpreter environment (default identifier is “name” but anything can be used in argument of Write command and numbers get added)

TOGETHER: .ls Draw some histogram namecycle->Draw() then .ls

now when we .ls we see the histogram has been moved from the file into memory but it is still associated with the file.

TOGETHER: _file0->Close()

Close the file. Oh no it’s gone. Can cause nasty crashes etc.

But, also is often fine way to work, especially with large objects like TTrees.

Introduce the two proper commands for getting files and getting the OUT of TFile and when in a script not interpreter mode. Also introduces pointer casting

TOGETHER: Read func1.cpp

TOGETHER: Explicitly write a TFile as a local variable

Make them write file open. Oh but it doesn't draw! Why? Need gROOT->cd and clone

TOGETHER: Add gROOT and cloning (func1b.cpp)

Finally show create output file

Can write into file explicitly by cd and hist→Write

TOGETHER: Modify to write by this method (func1b.cpp)

or can have file own histograms originally and do TFile→Write

INSTRUCTOR: Show off skipping middle step (func1c.cpp)

INSTRUCTOR: Talk through func2.cpp

INSTRUCTOR: Demonstrate opening TBrowser to look at content of file and play with GUI things

Very useful for most basic tasks, but switching from GUI to terminal to do more complex things can be messy, so often best to use scripts for anything complex

STUDENTS: Explore TBrowser and what can be done through GUI

Next - Summer School Material, expand TTree intro,