1/5/03

780.20 Monday Class

Gameplan:
- introductions — Furnstahl, TA?
- class philosophy/overview
- class logistics (step through info sheet)
- example projects
- background for 2062 session
- and Karl in SMA 207b/2082 working on session 1 sheet

To bring:
- digital camera, blank flappies, sign-up sheet
- books: Landau/Rez, Hirth-Jensen, Num. Rec. in C, Practical C++
- handouts: 780.20 info sheet, ch 2+3 handouts/extra excepts,
  sample code, 207b/2086 session 1, cpl handout, maple-sage,
  laptop + video projector (if available), extension cord

Before class:
- get pictures (if camera here)
- check for unregistered students
- check for unix accounts — sign up

Introduction + course history:
- Dick Furnstahl
  - nuclear theorist, low-energy QCD and nuclei
  - effective field theory methods
  - sufficiently complicated systems ⇒ numerical
    medium-scale rather than "large-scale" ⇒ sample:
    \( \text{MATLAB} \)
  - C/C++ and Mathematica mostly these days;
  - Fortran for many years; others: basic, perl, php,
  ...java, pascal, modula-3, lisp, maple, macsyma,reduce,...
  - usually unix (Linux these days)

Course history: first pass in fall 2002 (one quarter)

Still enduring/experimental
- originated from two motivations
  - my good students know how to program but not how to
    do computational physics
  - other good students in some sense
- long-term need for undergraduate comp. physics
1/21/03

- A computational physics course could be:
  - a physics course using computer as tool
  - a numerical algorithms course
  - a simulations course
  - a programming course
  - a guide to using environments like Mathematica or Matlab
  
- Plan: Aspects of all => sacrifice depth in favor of exposure to most important aspects using prototypical examples; practice in learning more on your own.
- Extreme version: one project per quarter (used with success elsewhere in advanced courses).
  - each person: do a project of their own

- Class background covers a wide range (from survey)
  - we'll have tutorial handouts as needed, but mostly learn as we go, a bit at a time
  - something for everyone, I think.
  - undergrads, first years, more advanced

**Carrying it out** (step through info sheet)
- Text: Last year used Computational Physics by Landau-Platz.
  - Hands-on, project oriented, breadth instead of depth (c.f. 6.077/6.047)
  - emphasis on errors, numerical libraries
- This year, use Martin Monk-Jensen's notes. Similar course in CSE.
  - C++, many examples and projects
- Other refs: Numerical Recipes, Practical C++ Programming.
  - many good online references

- Prerequisites:
  - mostly need to know the idea of a program
  - Linux you'll pick up quickly
  - editor doesn't matter—you've all used something
Comment on Material: on handout
- Contact info
  - Furnstahl - email always
  - TA?
  - Terry Bradley - computer stuff - help with cygwn, etc.
  - Tim Randels - can help with dual-booting if you fall
    in love with Linux

Schedule:
- first part of class: recap and preview/lecture/demo
- second part of class: hands on
  ⇒ probably efficient to do both in Smith 3016/3012

Office hours:
- drop by my office
- at Smith Lab 6-10pm M, W, Th ⇒ open lab? preference?

Grading:
- last year 5 problem sets ⇒ follow ups to class
  - project (see below)
  - really more like pass/fail, then grade

Project:
- everyone does a project - physics problem of interest
- could be thesis research, topic from textbooks, many examples
- combine tools and elements used in class
- analysis of correctness (how do you know the results are correct? how accurate are they?)
- scope/complexity dependent on your background
V5/03

- Look at 2.3 Implementation: Programming Concepts

- Go through area.cpp so same page ⇒ questions & comments
  - compare to Mathematica

- C source code ends in .c, e.g., area.c
- C++ " " " " .c, .cpp, .cpp, .cc
  ⇒ not standard. Here, .cpp
- Mathematica notebooks end in .nb ⇒ area.nb

- Compile area.c and links with gnu C++ compiler
  ⇒ g++ -o area area.cpp
  - program is called "area" ⇒ machine language (interpolated)
  - More generally, two steps:
    compile: g++ -c area.cpp ⇒ area.o
    link: g++ -o area area.o
    (other things)

- Linking reads in library code and puts it all together

- Good programming practices ⇒ cf. the code in handout
  - start with pseudo-code
    // read radius
    // calculate area of circle
    // \( \pi = 3.14159 \)
    // area = \( \pi \times \text{radius}^2 \)
    // print area

- Convert to code:
  1. elements of top documentation ⇒ see area.cpp
  - comments with // or // (first preferred)
  2. readability ⇒ indent
3. include files \Rightarrow f Mathematica packages
   \texttt{\textless; Graphics
   \hspace*{1cm} \texttt{\textless; Namespave analog in Mathematica \Rightarrow look-}
   \hspace*{1cm} \texttt{\textless; For now, use this as a template}

4. const double
   \texttt{\textless; double precision, single precision is Float}
   \texttt{\textless; precision refers to number of digits kept}
   \texttt{\textless; Referred to \texttt{\#define PI 3.14159} in representation (important that it is \texttt{\#define})}
   \texttt{\textless; More later.}

5. main routine with \texttt{f} \texttt{f}'s

6. declare all variables \texttt{\textless; usually when first used (of C)}
   \texttt{\textless; Use good variable names}

7. C vs. C++ input/output

8. calculation: why radius * radius instead of \texttt{\texttt{Pi}}?
   \texttt{\texttt{For the newer to C/C++ \Rightarrow could you change to calculate a volume?}}

Mathematica "program"
\texttt{area[\texttt{\texttt{radius}}]} \texttt{\texttt{\Rightarrow PI + radius^2}}
Representation of numbers — "Floating Point"

- Classic computer nerd T-shirt: "There are 10 types of people in the world..."
- Those who understand binary and those who don't

Unique, well-defined representation for "floating point" numbers:

\[(\text{any number}) = (-1)^{\text{sign}} \times (\text{base}^2 \times \text{mantissa}) \times 2^{\text{exponent bias}}\]

- Base 10 with bias = 5

\[-\frac{14}{3} = \frac{-1.3333}{3} = (-1)^{\text{sign}} \times (\text{mantissa}) \times 10^{\text{exponent bias}}\]

- Largest, smallest, precision:

  - Largest: \[3.4 \times 10^{38}\]
  - Smallest: \[2.9 \times 10^{-38}\]
  - Precision: \[6-7\] decimal places

- Single precision float: \(4\) bytes \(\Rightarrow\) 32 bits = 1 sign + 23 mantissa + 8 exponent

- Double: \(10^{32}\) to \(10^{38}\) decimal places

⇒ Try it out in practice!

- What about a simple decimal: \(\frac{1}{5} = \frac{2}{10}\)

\[101 + 0.11_2 = 1 + 2^{-1} + 2^{-2}\]

- Binary floating point:

\[0.1 \times 2^{-1} + 0.111_2 \times 2^{-2}\]

\[0.1 \times 2^{-1} + 0.111_2 \times 2^{-2} = 0.605\]

- How to find bias in Mathematica:

\[\text{FlushedRight} = 1000\]
1/5/04
* In 2002... upgrade to Fig. 3.1

Program Development

shell

widths

kernel

hardware

device

op

syscall

topology

xwindow

Vi

emacs

mail

bush

tcsh

g++ make gdb

- try logging into nighthawk and demoing dft_trap_menu -