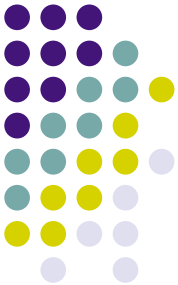
# ATSC 212
## Earth & Atmospheric Science Intro Computing Lab

Roland Stull , George Hicks II
rstull (at) eos.ubc.ca ghicks (at) eos.ubc.ca
Physical scientist Computer scientist





## Welcome

- Introduction of instructors & TA.
  TA: Jesse Mason, jmason (at) eos.ubc.ca
- Introduction of students.
- Survey of existing student programming. backgrounds.
- Marking philosophy (not curved).
- Help your classmates.

# Get your lab account NOW

- Either bring your own laptop to lab, or
- Use the existing in-lab computers.
  If you don't already have an account on the EOSC lab computers, then:
  - After class, Go to room 2604 Copp Bldg.
  - Ask for Ms. Carol Leven (an EOSC secretary)
  - Pay her $25
  - Bring your receipt back to me.
  - We can activate your account as soon as you show us the receipt.



# If you use your own laptop

- Please load the following software onto your computer. See our "resources" web page for links:
  - ***KompoZer*** (PC, Mac, or Linux) for web authoring.
  - ***FileZilla*** (PC) or ***CyberDuck*** (Mac) for transferring files from one computer to another.
  - ***NX client*** (PC only), to use your laptop as a terminal to a remote Linux server. (Macs already have a ***Terminal*** program built in, but it might be easier to use the NX Client for Macs.)
  - ***VPN*** virtual private network, provided by UBC IT Services, to allow you to log in to our servers.

# Meetings

- Dates: Tuesdays & Thursdays (8 weeks)
- Times: Officially 11:00 am - 12:15 pm
- Place: Computer Lab, room 203 in Earth & Ocean Sci.-main (EOSM)



# Schedule

**Week. Topic (instructor)**

1. Welcome (RS & GH) & Linux (GH)
2. Linux & VI (GH)
3. emacs & FORTRAN (RS)
4. FORTRAN (RS)
5. C (GH)
6. C (GH)
7. html (GH)
8. html (GH) & Conclusion (RS & GH)

Assumption: You have already learned how to Program in Python, MatLab, and/or Excel.

# Goals

- To expose you to different programming languages used in science & engineering.
- To increase your chances of getting a job or being accepted into grad school.
- To allow you to list on your resume that you've written some code in these various languages.
- To give you the confidence (and exposure to resources) that you need to learn more about these languages on your own.



# Typical Class Agenda

## Tuesdays

- Lecture by instructor (5 - 45 min)
- Programming demonstration by instructor, with students following along (5 - 30 min)
- Students extend the programming, with instructor following along (5 - 30 min)
- Students finish the in-lab programming, with instructor & TA helping when asked (5 - 30 min)
- Preview lecture by instructor of topics for next week
- Readings & HW programming assigned (due next week)

## Thursdays

- Finish any remaining lecture by instructor.
- Finish lab assignment.

# Required Resources

- Custom Course Material pack. (UBC Bookstore)
- Account on EOSC computers, or your own laptop.
- Campus-wide log-in & VPN. (Allows you to access our machines via the UBC Library firewall.)



# Optional References


- Cooke, C., 2007: KompoZer User Guide. A pdf file from http://www.charlescooke.me.uk/web/kz_user_guide-ss.pdf
- D. S. Ray, 2002: *Mastering HTML and XHTML*, Sybex. 1107pp. ISBN 0782141412.
- B.W. Kernighan and R. Pike, 1984: *The UNIX Programming Environment*. Prentice Hall. 357 pp. ISBN 0-13-937699-2 or ISBN 013937681X.
- B.W. Kernighan and R. Pike, 1999: *The Practice or Programming*. Addison-Wesley. 267 pp. ISBN 0-201-61586-X.
- S.J. Chapman, 2004: *Fortran 90/95 For Scientists And Engineers, 2nd Ed*, McGraw Hill Canada. ISBN 0-07-282575-8.
- S.J. Chapman, 2008: *Fortran 95/2003 for Scientists and Engineers, 3rd Ed*. McGraw Hill. ISBN 978-0-07-319157-7.
- B.W. Kernighan and D.M. Ritchie, 1988: *The C Programming Language, 2nd Ed*. Prentice Hall Software Series. ISBN 0-13-110362-8.
- S. Holzner, 2004: *Perl Core Language Little Black Book, 2nd Ed*., O'Reilly. ISBN 1-932111-92-1.
- Reese, G., R.J. Yarger, T. King, 2002: *Managing & Using MySQL, 2Ed*. O'Reilly.
- Press, Teukolsky, Vetterling, Flannery: *Numerical Recipes in ___: The Art of Scientific Computing*. Cambridge Univ. Press. ___ = FORTRAN, C, C++, and many more

# Programming Realities

- You are a physical scientist or engineer.
- You are programming for yourself.
- Your goal is to solve a physical problem, or to simulate the workings of nature.
- Once the program is successfully written, debugged, and tested, you will use it only once, or a few times.
- The most expensive part of this activity is
  - You (writing the program)
  - Not the computer (running the program)
- Once the program works, you will usually change it before you run it again.



# Programming Philosophy - 1

- Think of the whole scientific problem before you start writing code. (Think before you write.)
- Program from the top down. (write the main program first, with stubs for your "helper" functions)
- Test your code incrementally, as you write it.
- Use Version control.
- Document each line as you write it, using in-line comments. Include physical units in the comments.
- Add full-line comments as header paragraphs
  - Explain what aspects of physics your code describes.
  - Cite main eqs. from journal papers or books.

# Programming Philosophy-2

- Write clear, simple, logical code.
- Use strong typing (declare all variables).
- Don't use cryptic variable names.
- In your type declarations, indicate the physical units via in-line comments.
- Initialize variables during declaration.
- Test intermediate & extreme values against known outcomes or hand calculations.

Namely, follow the Rules in the Appendix of Kernighan & Pike (included in the Custom Course Materials).



# Programming Philosophy-3

- "*Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.*"
  - Brian W. Kernighan
- Other good quotes on Software Design for Maintainability: http://gael-varoquaux.info/blog/?p=136

# Reading Assignments

- Week 1: Hayes, B., 2006: The semicolon wars. *American Scientist*, **94**, 299-303. (in course pack)
- Week 1. Read Kernighan & Pike chapters in the following order, (in the course pack) Focus on concepts, not code:
  1. Epilogue
  2. Appendix - Collected Rules
- Other handouts (assigned later)



# Lab Policy & Security

- Scheduled classes/labs have priority over drop-in users.
- This room is reserved for ATSC 212 from 11:00 - 12:30 on Tuesdays & Thursdays, so you can stay here until 12:30 pm to work on your assignments.
- Always protect your password.
- Always log out from the lab computer when finished, but leave it turned on.
- Report any hardware problems to Sukhi .

# Electronic-Media Ethics

- See University Policy 104, on **Responsible Use of Information Technology Facilities and Services**
- http://www.universitycounsel.ubc.ca/policies/policy104.pdf
  - 2.1. Users must
    - 2.1.1. preserve the privacy of data to which they have access;
    - 2.1.2. respect the privacy of others by not tampering with e-mail, files, or accounts they use; and
    - 2.1.3. respect the integrity of computing systems and data.
  - 2.2. For example, users must not: intentionally develop programs or make use of already existing programs to harass other users, infiltrate a computer or computing system, damage or alter the components of a computer or computing system, gain unauthorized access to other facilities accessible via the network, or inappropriately use the telephone system.



# Don’ts

- **6. Examples of Illegal Uses**
  - 6.1. The following are representative examples only and do not comprise a comprehensive list of illegal uses:
    - 6.1.1. uttering threats (by computer or telephone);
    - 6.1.2. distribution of pornographic materials to minors;
    - 6.1.3. child pornography;
    - 6.1.4. pyramid schemes; and
    - 6.1.5. copyright infringement.
- **7. Examples of Unacceptable Uses**
  - 7.1. The following are representative examples only and do not comprise a comprehensive list of unacceptable uses:
    - 7.1.1. seeking information on passwords or data belonging to another user;
    - 7.1.2. making unauthorized copies of proprietary software, or offering unauthorized copies of proprietary software to others;
    - 7.1.3. copying someone else’s files, or programs, or examining such information unless authorized;
    - 7.1.4. attempting to circumvent computer security methods or operating systems (e.g. subverting or obstructing a computer or network by introducing a worm or virus);
    - 7.1.5. using University-provided computer accounts for commercial purposes such as promoting by broadcast non-educational profit-driven products or services;
    - 7.1.6. intercepting or examining the content of messages, files, or communications in transit on a voice or data network;
    - 7.1.7. interfering with the work of other users of a network or with their host systems, seriously disrupting the network (e.g. chain letters or spamming), or engaging in any uses that result in the loss of another user’s files or system; and
    - 7.1.8. harassing or discriminatory telephone messages.

## Help

- FAQs on course web page
- Online users manual ("man" pages)
- Online Wikipedia
- Printed books in lab & library
- Email to TA: Jesse Mason
  jmason (at) eos.ubc.ca
- Email to Instructors: (see title page)



## Summary. You will learn:

- Good programming practices
- Common tricks and pitfalls in scientific programming
- Hands-on experience with
  - Tools: VI, Emacs
  - Compiled Languages: FORTRAN, C
  - Operating System: UNIX/Linux
  - Web authoring: html

Any Questions so far?

# Intermission

- Health break for 5 minutes
- After which you
  - Work with George Hicks and Jesse Mason to get the software installed on your computers, or
  - Learn how to access the software on the computers in the lab room
- Start the UNIX/Linux section (instructor is George Hicks).