Computational Physics

Introduction

01/06/2009
Course Overview
Structure
Goals
Computing Resources

Brief Introduction to Linux/Unix

Unix Basics
Computational Physics

A First Course in Computational Physics and Object-Oriented Programming with C++

HCB 219: Tuesday, Thursday 2:00 – 3:15

Prof. Volker Credé

Office: 206 Keen Building
Office Hours: Wednesday 3:00 – 4:00 PM
(Other times are possible)
Phone: 644-2423
Email: crede@fsu.edu
Syllabus

Course WEB Address:

http://hadron.physics.fsu.edu/~crede/TEACHING

or

Blackboard ➜ http://campus.fsu.edu

Textbook:

A First Course in Computational Physics and Object-Oriented Programming with C++

by David Yevick
First Day Attendance

Log on to course website (Blackboard):
http://campus.fsu.edu

Use “First Day Attendance” tab and enter the passcode “Comp2009”
Course Organization

Lab Classes: HCB 0219
Tuesday & Thursday 2:00 to 3:15

- **Discussion of homeworks**
  (Homeworks are due in class on Tuesday)

- **Projects**
  (Projects are due at the end of the Thursday class)

Homework Assignments
Grading of the Course

Grades will be based upon the successful completion of assignments (homeworks), exercises, and projects.

- **Homework Assignments** 30%
  
  Homework problems are typically from the course text.

- **Exercises and Projects** 70%
  
  - In-class computational and programming exercises
  
  - Extensive exercises utilizing scientific programming techniques to model physical systems

- **No tests or final exam in this course**
Course Assistant

Darren McGlinchey

Office: 120 Collins Research (NRB)
Phone:
Email: dcm07e@fsu.edu
Outline

1. Course Overview
   - Structure
   - Goals
   - Computing Resources

2. Brief Introduction to Linux/Unix

3. Unix Basics
Course Overview

- Introduction to Linux/Unix programming tools
- Programming in C++ and OOP
- Plotting & visualization packages and web resources
- Numerical limits in computing
- Finding roots of equations
- Code management techniques
- Numerical differentiation & integration
- Multidimensional & Monte Carlo integration
- Analyzing data sets
- Introduction to batch & parallel programming
The programming language for this course is Object-Oriented C++.

About 1/3 of this course will focus on C++ programming basics (part 1 of text).
Computing Resources

Physics Linux Cluster
- comphy.fsu.edu
- Everyone gets a user account

HCB 219 Computers
- ssh & vnc connection
Outline

1. Course Overview
   - Structure
   - Goals
   - Computing Resources

2. Brief Introduction to Linux/Unix

3. Unix Basics
Operating System: Components

- The Operating System
  - Kernel
  - Shell
  - Programs & Commands

- The File System
  - Paths
  - Tree Structure
  - Directories, Files, and Inodes

- Desktop Environments
The Operating System

- **Kernel**

  The kernel is the central component of most computer operating systems (OS). Its responsibilities include managing the system’s resources (the communication between hardware and software components).
The Operating System

- **Kernel**
  A kernel connects the application software to the hardware of a computer.

- **Shell**
  - Command interpreter providing a layer between the operating system and the user
  - Several shells are available
    - sh, csh, ksh, tcsh, bash, etc.
    - Each shell includes a programming or shell language.

- **Programs**
  - Unix provides several hundred utility programs.
  - Shell scripts also provide utility.
The File System

Tree Structure

File-System Components: Directories and Files

- Path Names: /home/tux/test.c  "absolute"
The File System

Tree Structure

File-System Components: Directories and Files

- Path Names: ./tux/test.c  "relative" from /home
Linux Desktop Environment

- **Operating System (OS)**
  - Linux/Unix ⇔ DOS/NT

- **Desktops**
  - KDE, GNOME ⇔ Windows (XP), MAC OS X
Programs and Commands
Getting Started

- All students have `comphy` user accounts.
- HCB 219 Computers
  - VNC Viewer (ssh to comphy and run a vnc server)
  - ssh terminals (wo/ X windows)

Keen Building, 3rd floor
SSH Connection to COMPHY

On the lab computer:

- Run ssh program
  - Connect to comphy.fsu.edu
  - Username <LASTNAME>
  - Password <USERNAME1234>

- Change your password
  - Use the PASSWD command

- Don’t forget your username and new password!
Starting a VNC Session

VNC stands for Virtual Network Computing. It is remote control software which allows you to view and also to fully interact with one computer desktop (the "VNC server") using a simple program (the "VNC viewer") on another computer desktop anywhere on the Internet. The two computers don’t even have to be the same type, i.e. you can use VNC to view a Windows Vista desktop at the office on a Linux or Mac computer at home.
Starting a VNC Session

On `comphy.fsu.edu`:

- Run `VNCSERVER`:#
  - Define a password
  - Change password via `VNCPASSWD` command

- **VNCserver Documentation**
  - Use the Linux manual pages via `MAN VNCSERVER`.
  - See online manuals at [www.TightVNC.org](http://www.TightVNC.org).

On your computer (Windows, MAC, Linux, ...):

- Connect via VNCviewer to host `comphy.fsu.edu`:#
VNC@home

Get and install VNC viewer

• [http://www.realvnc.com](http://www.realvnc.com)
• [http://www.tightvnc.com](http://www.tightvnc.com)

On comphy.fsu.edu:

1. Edit the file `.vnc/xstartup` by typing:
   ```
   NANO ~/.vnc/xstartup
   ➔ Uncomment the first lines as suggested!
   ```
2. Kill the vncserver: `VNCSERVER -KILL` :
3. Restart: `VNCSERVER (-GEOMETRY 1280x1024)` :
Outline

1 Course Overview
   Structure
   Goals
   Computing Resources

2 Brief Introduction to Linux/Unix

3 Unix Basics
# Common File System Navigation

## Directories

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd &lt;dir/&gt;</td>
<td>change directory</td>
</tr>
<tr>
<td>pwd</td>
<td>print working/current directory</td>
</tr>
<tr>
<td>mkdir &lt;name&gt;</td>
<td>make new directory</td>
</tr>
<tr>
<td>rmdir &lt;name&gt;</td>
<td>remove empty directory</td>
</tr>
<tr>
<td>ls -dl &lt;dir/&gt;</td>
<td>list directory properties</td>
</tr>
</tbody>
</table>
## Common File System Navigation

### Files

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
</table>
| cp      | copy `<file1>` to `<file2>`
|         | copy `<files>` to `<dir/>`  |
| mv      | move `<file>` to `<dir/>`
|         | rename `<file1>` to `<file2>` |
| ls      | list files             |
| ls -l   | list files with property info |
| ls -a   | list invisible content  |
# Unix Wild Cards

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>match any single character</td>
</tr>
<tr>
<td>*</td>
<td>match any size string</td>
</tr>
<tr>
<td>[abc]</td>
<td>match any enclosed character</td>
</tr>
<tr>
<td>[a-f]</td>
<td>match any character in range</td>
</tr>
<tr>
<td>![abc]</td>
<td>match all but enclosed characters</td>
</tr>
<tr>
<td>~</td>
<td>current user home directory</td>
</tr>
<tr>
<td>~user</td>
<td>home directory of a user</td>
</tr>
</tbody>
</table>

**Examples:**
## Unix File Redirection

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>redirect to standard output</td>
</tr>
<tr>
<td>&gt;&amp;</td>
<td>redirect to standard error</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>append to standard output</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>&amp;</td>
</tr>
<tr>
<td>&lt;</td>
<td>input redirection</td>
</tr>
<tr>
<td>&lt; &lt; String</td>
<td>read from standard input until “String” is encountered as the only thing on the line</td>
</tr>
</tbody>
</table>

**Examples:**
- `cat file1 file2 > file3`
- `cat file1 > file3`
- `cat file2 >> file3`
### Special Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>command separator</td>
</tr>
<tr>
<td>&amp;</td>
<td>run command in background</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>run next command upon success</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>command'</td>
<td>execute command first &amp; substitute result</td>
</tr>
<tr>
<td>\</td>
<td>escape the following character</td>
</tr>
</tbody>
</table>

**Examples:**

- `cat file1 > file3 ; cat file2 >> file3`
- `emacs file1 &`
- `grep string file || echo "string not found"`