Announcements

• HW01 is due this Thursday, electronically by noon.

• Start compiling lab and HW programs using this command:
  
  
  "g++ -Wall -o myprogram myprogram.cpp"

• My office hours:
  – 3:30-5pm Tuesdays in Room 022-C (our computer lab)

• TA office hours
  – In Room 022-C
    • Mondays 7-9:30pm
    • Tuesdays 4:30-6:30pm
    • Wednesdays 7-9:30pm
Review and Outline

• Recall last time:
  – Introductions
  – Computer parts
  – Operating systems, kernel, shells
  – The Linux command line, shells, simple commands

• Today:
  – Remote connections, both shell and graphics
  – Intro to programming languages and compilers
  – C program structure
  – Defining simple variables and doing arithmetic
  – Formatted input/output
The  Shell

**What's a Shell?**

The Shell intermediates between the user and the operating system. It accepts commands from the user, uses the operating system to execute them, and then returns results to the user.

Two major types:
- Command line interface (CLI)
- Graphical user interface (GUI)
The Command Line Interface

• More emphasis in Linux than in other OS’s in user-interactions executed through the **command line** rather than through graphical user interface (GUI) than eg Windows

```bash
[phys2660@node5 ~]$ ls -lrt
total 9748
drwxr-xr-x  4 phys2660 phys2660    4096 Feb 22  2007 lib
drwxr-xr-x  4 phys2660 phys2660    4096 Apr  4  2007 local
drwxr-x---  2 phys2660 phys2660    4096 Aug 10  2007 Desktop
drwxr-xr-x 11 phys2660 phys2660    4096 Mar 20  2008 usr
```
Commands and Arguments

ccn4g@galileo /home/ccn4g] ls
btag/ Desktop/ mail/ phys2660_spring2012/ phys5630_fall2010/ public/
public_html/ ROOT_intro/
ccn4g@galileo /home/ccn4g] ls -l
btag/
Desktop/
mail/
phys2660_spring2012/
phys5630_fall2010/
public/
public_html/
ROOT_intro/
ccn4g@galileo /home/ccn4g] ls -lrt
total 32
drwxr-x--- 2 ccn4g ccn4g 4096 Aug 10  2007 Desktop/
drwx------ 2 ccn4g ccn4g 4096 Feb 15  2010 mail/
drwxr-xr-x 2 ccn4g ccn4g 4096 Feb 16  2010 ROOT_intro/
drwxr-xr-x 2 ccn4g ccn4g 4096 Feb 16  2010 btag/
drwxr-xr-x 2 ccn4g ccn4g 4096 Aug 24  2010 public_html/
drwx-rwrxrwx 2 ccn4g ccn4g 4096 Sep 19  2010 public/
drwxr-xr-x 5 ccn4g ccn4g 4096 Sep 19  2010 phys5630_fall2010/
drwxr-xr-x 3 ccn4g ccn4g 4096 Jan 23 14:46 phys2660_spring2012/

Look at the different output of the program ‘ls’ when passing different arguments
Command Line Interface

• Virtues of the command line:
  – Can narrow results with wildcard expressions using “*”
  – Text interactions are easily reproduced user-to-user or session-to-session (command history) whereas it is more awkward to describe graphical interactions
  – Allow for aliasing commands to more familiar, user-defined choices
  – Establishment of environment variables for convenient consistent functions
  – Graphical interactions are nice but typically of limited use
  – Linux has graphical interfaces but they are inconsistent among different Linux variants
  – Not every computer has graphical tools installed, especially if it was built for performance – but there will ~always be command-line interactions
  – Many simple things do not need a fancy graphical output (think ‘ls’ instead of Windows Explorer..)
Illustration of the Linux Filesystem

The Linux Filesystem:

Here's a graphical representation of a highly simplified Linux directory tree. One of the basic principles of Linux (and other varieties of Unix) is that there's only one directory tree. Everything lives somewhere under the “/” (root) directory.

This is unlike Windows, for example, where each device has a separate directory tree. In Windows we have a directory tree on drive C:, a different one on drive D:, and so on. Under Linux all files on all devices show up somewhere in the same directory tree, with “/” at its top.

This file is “/home/bryan/file.txt”

et cetera...
Navigating Filesystem and Special Directories

• Think of the filesystem as a nested collection of folders, a user has a home location and a current location

• How to tell your current location: `pwd`

  ```
  [phys2660@node1 phys2660_spring2012]$ pwd
  /home/ccn4g/phys2660_spring2012
  ```

• Change to a different directory: `cd`

  ```
  [phys2660@node1 phys2660_spring2012]$ ls
  prelab_01
  [phys2660@node1 phys2660_spring2012]$ cd prelab_01/
  [phys2660@node1 phys2660_spring2012]$ ls
  
  [phys2660@node1 phys2660_spring2012]$ cd prelab_01/
  ```

• A user’s “home” directory:

  ```
  ccn4g@galileo /home/ccn4g/phys2660_spring2012] echo $HOME
  /home/ccn4g
  ```
The PATH Environment Variable

ccn4g@galileo /home/ccn4g/phys2660_spring2012] echo $PATH
/home/phys2660/usr/bin:./usr/lib/qt-3.3/bin:/usr/kerberos/bin:
.:./local/bin:/cluster/bin:/common/bin:/bin:/usr/bin:/usr/games:
/usr/X11R6/bin:/usr/openwin/bin:/common/CERN/pro/bin:
/common/lib/scilab/bin:/common/lib/LASSP/bin:/usr/NX/bin:
/opt/real/RealPlayer

- Typing commands = running a program
- Compiled version of program can be anywhere on the disk
- PATH variable tells your shell what locations within the context of the filesystem to look for these executables

- For instance, most common Linux commands reside under /bin or /usr/bin
Convenience: Command History

- On galileo, one can use up arrow to access commands in your history
Convenience: Manual Pages Within Linux

ccn4g@galileo /home/ccn4g/phys2660_spring2012] man ls
LS(1)        User Commands
NAME
   ls - list directory contents
SYNOPSIS
   ls [OPTION]... [FILE]...
DESCRIPTION
   List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort.
   Mandatory arguments to long options are mandatory for short options too.
   
   -a, --all
       do not ignore entries starting with .
   
   -A, --almost-all
       do not list implied . and ..

• good programs have easy-to-read manual pages, called man pages. Use: man <commandname>
### Some Common Linux Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ls</code></td>
<td>List the contents of a directory.</td>
</tr>
<tr>
<td><code>pwd</code></td>
<td>Show the name of the current directory.</td>
</tr>
<tr>
<td><code>cd</code></td>
<td>Change the current directory.</td>
</tr>
<tr>
<td><code>less</code></td>
<td>Show the contents of a file, one page at a time.</td>
</tr>
<tr>
<td><code>more</code></td>
<td>Show the contents of a file, one page at a time.</td>
</tr>
<tr>
<td><code>cp</code></td>
<td>Copy a file.</td>
</tr>
<tr>
<td><code>mv</code></td>
<td>Move (rename, relocate or both) a file.</td>
</tr>
<tr>
<td><code>rm</code></td>
<td>Delete (remove) a file.</td>
</tr>
<tr>
<td><code>mkdir</code></td>
<td>Make a new directory.</td>
</tr>
<tr>
<td><code>rmdir</code></td>
<td>Delete (remove) a directory.</td>
</tr>
<tr>
<td><code>man</code></td>
<td>Show docs (manual pages) for a command.</td>
</tr>
<tr>
<td><code>ln</code></td>
<td>Make a link to a file.</td>
</tr>
<tr>
<td><code>cat</code></td>
<td>Spits out the concatenated contents of one or more files, without paging.</td>
</tr>
<tr>
<td><code>touch</code></td>
<td>Change the timestamp on a file, or create an empty file.</td>
</tr>
<tr>
<td><code>which</code></td>
<td>Find a command in the search path.</td>
</tr>
</tbody>
</table>
Editing Files

• One of the main tasks you will do is edit human-readable files

• Many tools available to you to do this:
  – nano (you’ll use this in lab01)
  – pico (similar to nano)
  – nedit
  – emacs (lab01)
  – See this wiki page for some more discussion
Case Sensitivity:

Important Note: when typing commands, file names, etc...

Linux and C/C++ are case sensitive

Case Sensitivity

So, for example:

This is not the same as this,
VELOCITY is not the same as Velocity or velocity,
MyFile.dat is not the same as MyFile.DAT

For best results, stick to all lower-case unless there's a good reason to do otherwise.
Local vs. Remote Computing
The “Galileo” Cluster:

In this class, we'll do our programming on a Linux cluster called “Galileo”. Galileo doesn't have any keyboards or monitors connected to it. So how do we use it?

[Image of the Galileo web page]

http://galileo.phys.virginia.edu/
Remote Shells

- So far we have tacitly overlooked one key component:
  - What if I am not sitting in front of the specific computer / Linux machine I want to work on?
  - How do I establish a connection to that machine?
  - Use the network!

- Remote shell access:
  - We will use a program called ssh
    - Secure Shell
    - Replaced an old lovable program called telnet, which, in terms of security, had more holes than swiss cheese! Totally Unencrypted
    - Tip: Never ever type the command telnet or even consider typing it
  - FYI There is no ssh version for Windows by default but you can install a Windows version for free. See Software for home on the class home page.
Galileo: Where We Will Do Our Work

- The computer / Linux “machine” where we will do all our work is in the physics department and goes by the name galileo
- galileo is actually the head node of a cluster of computers running Linux

```bash
d-172-25-103-246 /Users/neu]
ssh -Y -l ccn4g
galileo.phys.virginia.edu
ccn4g@galileo.phys.virginia.edu's password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
Last login: Mon Jan 23 14:08:26 2012 from
d-128-100-66.bootp.virginia.edu
   .cshrc sourced
ccn4g@galileo /home/ccn4g]
```
A Note on Our Computer Lab

Our Computer Lab:

Room 022

Lab Computers

Galileo Cluster

Room ??
The Computers in the Lab

- The desktop machines in our computer lab run Lubuntu v12.04, kernel v3.2.0-24-generic
  - a “lightweight” Linux operating system based on Ubuntu.
  - Ubuntu is a popular (most popular?) type of Linux OS distributions with its own desktop environment.
  - There are several such distributions popular in the community today, Ubuntu is just one

The actual nodes on galileo run a Linux distribution called CentOS, release 5-11. The kernel is v 2.6.18-407.el5
Use ‘lsb_release –a’
Remote Graphics

• **X Server**: running on your local computer

• **X Client**: program to display graphics from any other program

• Can be on different computers!

Under Linux, graphical programs use a protocol called “X” to draw images on your screen. X is “network transparent”, meaning that X clients don’t care whether they’re running inside your computer or on a different computer, half a world away.
Connecting to galileo from other locations/laptops

• It would be nice to connect to galileo to do your work from say, your laptop while sitting in the library for instance.

• Many hints on how to do this are located here: http://faculty.virginia.edu/comp-phys/phys2660/wiki/doku.php?id=software:sw_main

• If you run Windows
  – I recommend installing and using putty with Xming
  – Another option for Windows: NXClient for speedier graphics connections. See me if interested – might be better than putty +XMing

• If you run Mac OS X
  – ssh is installed but you still need an X Server to see graphics. See ssh and X11 for Mac
Computer Programming in C: The Beginnings
Programming Languages

• As we mentioned earlier, the hardware inside a computer does not speak a human-readable language

• But we as humans want to harness the power of the computer as a tool to solve problems

• Two options:
  – We communicate directly with computer using its language
  – We write programs in something more familiar to us, and figure out how to translate

• Option 2 is preferred!
Dennis Ritchie:
- defined the relatively simple syntax of C as part of original Unix R&D
- wrote the first compiler for C programs

Standard simple language meant people could easily write compilers for any CPU architecture AND users could easily write new programs
Compilers:

A programming language is like a human language. It has a vocabulary (reserved words with special meanings) and a syntax or fixed rules for using the words.

The programming language is a high-level tool to define the steps your computer must take to solve a problem.

In the end, a program in any language you choose must be reduced (compiled / assembled / interpreted) to a set of instructions the CPU can understand in order for it to function (machine code).

In general, these instructions will differ from one type of CPU to another, but most computers today use CPUs that understand a common set of instructions called “IA-32”.

printf (“Hello world!”);

457f 464c 0101 0001 0000 0000 0002 0003...
Our compiler: g++

GNU C/C++
In this class, we'll be using g++, a compiler originally written by Richard Stallman as part of his GNU Compiler Collection project:

Frontends
- g++ (C++)
- gcc (C)
- gfortran
- gcj (Java)
- others...

Backends
- IA-32
- Alpha
- ARM
- m68k
- others...

g++ is just a program, and you invoke it from the command line like this:

```
~demo> g++ -o myprogram myprogram.cpp
```

This would cause g++ to read the file “myprogram.cpp” and produce the output file “myprogram”, which is an executable binary file. You could then run your program by typing “myprogram” at the command line.

28
The Structure of a C Program:

A C or C++ program is built up out of units called “functions”. Each program must have at least one function, called “main”. The large-scale structure of a C program consists of a statement like the one below, defining what this “main” function should do.

In the example below, the “main” function doesn't have any arguments, but we'll see later that it can do so.

```c
int main() {
    The body of the program goes here.
    return(0);
}
```

The body of the function is delimited by braces.

The function returns a value. In the case of “main”, the operating system uses this value to determine whether the program completed successfully. By convention, non-zero means an error occurred.
The C Language: An Example

A Simple C Program:

```c
#include <stdio.h>
#include <math.h>

int main() {
    int a = 2;
    int b = 2;
    int c;
    double d;

    printf("Hello, world!\n");

    c = a*b;
    printf("The value of c is %d\n", c);

    d = sqrt(a);
    printf("The square root of %d is %f\n", a, d);

    return(0);
}
```

- Commonly-used statements can be stored in external "header" files and re-used in other programs.
- Each part of your program is a function. At the highest level is a special function named "main".
- Every statement must end with a semicolon.
We’ll pick up from here next time.

See you then.