Using the Lab

http://people.sc.fsu.edu/∼jburkardt/isc/week01/
lecture_02.pdf

....

ISC3313:
Introduction to Scientific Computing with C++
Summer Semester 2011

....

John Burkardt
Department of Scientific Computing
Florida State University

Last Modified: 09 May 2011

Introduction to Scientific Computing with C++

INTRO:

Today we will not have a regular lecture, because your instructor is out of town on a business trip.

Today we will concentrate on helping you explore the lab computers, and the software programs installed there.

We will have you go through some simple exercises:

- logging in;
- setting up your directory;
- downloading a file using the browser;
- editing a C++ program;
- compiling a C++ program;
- running an executable program;

We end with an in-class exercise that you must complete for credit.

LOGIN: Local Access

Students in this class can log into any computer in this lab.

You may also use the hallway computers on the fourth floor of Dirac Science Library, from 8-5 every weekday.

The lab computers and hallway computers have a single, shared file system. It doesn't matter which machine you log into; you will always have access to the same set of files.

To log in, you need to use your FSU ID and password.

Try it now! If you are unable to log in, please let us know!

LOGIN: Remote Access

You may find it useful to be able to move files between the lab machines and your home computer.

You can access your files from your computer at home or another lab, using programs such as

- **ssh** for interactive use;
- **sftp** to transfer files.

For instance, if your computer uses the Unix system, and you run the ssh program from a terminal window, you can type:

```
ssh pamd.sc.fsu.edu
```

which will let you access your files on our system.

Windows users can get the free **putty** program for remote access.

Introduction to Scientific Computing with C++

DIRECTORIES: The Visual Interface

The lab computers run a version of the Linux operating system, which includes a convenient visual interface.

You will see items on the top menu bar, including:

- **Applications**,
  - **Accessories**
    - **Terminal** for command line interface;
    - **Text Editor** to create and modify files.
  - **Programming**
    - **KDevelop C/C++** an IDE for C++ programming;
- **Places**
- **System**
  - **logout** from the system;

DIRECTORIES: Use the Browser to Get a File

You will also have icons on the left side, such as:

- **Web browser**
- **Home** for storing and organizing files;
- **Trash** for deleting files.

Double click on the browser to start it, then go to
http://people.sc.fsu.edu/~jburkardt/isc

Move to the **Week 1** subdirectory.

Select the **hello.cpp** program.

Then, from the browser's **File** menu, choose **Save page as**. The default name and location are fine. Choose **OK**. The file will be saved to the Desktop.

Choose **Quit** from the browser's menu.

DIRECTORIES: Store the File in a Directory

Double click on the **Home** icon, which should open up your home directory. There's probably almost nothing there right now. We want to put your file there, but in a separate subdirectory.

From the **Edit** menu item on your Home Directory window, choose **Create New...** and then **Folder**. When asked for a name for the folder, type **week1**. You should now see the image of a folder in your Home Directory window.

Locate **hello.cpp** on your Desktop, and drag to the **week1** folder.

When you "let go" of the file, you will be asked if you want to **copy** or **move** the file. Choose **move**.

Click on the **week1** folder, so you see the file **hello.cpp** there.

This suggests some of the ways in which the visual interface can be used to view your folders and files, to create new folders and to move files around.

DIRECTORIES: Using the Command Line Interface

While the visual interface is easier to use, there are times when it is necessary to use what is called the command line interface.

This involves running the Terminal program, which opens a command window. You type your commands in the window. The terminal program uses the same file system as the visual interface. What the visual interface calls *folders* we will now call directories.

You start in your home directory. You can use commands to move to a new directory, to get a list of files in a directory, to create new directories, and to move or copy files.

The commands we will use are known as **unix** commands; these commands are very common across almost all computer systems these days, except for Windows PC's.

DIRECTORIES: Open a Terminal in Home Directory

The terminal program is available from the **Applications/Accessories** menu.

The terminal always has a *present working directory*, that is, the directory (or folder) where it is working. When the terminal program begins, it starts in your home directory. Because everything is done with words, not pictures, your home directory can be identified by a long complicated name. We can always ask the terminal program to give the name of its present working directory with the **pwd** command:

```
pwd
  /panfs/panasas1/users/jburkardt
```

Luckily, we rarely need to type this long name in. And there is a shortcut name for your home directory: **$HOME**.

DIRECTORIES: Listing Directories and Files

Since we don't have a visual interface, we need a command to "see" what's in the current directory. The **ls** command will "list" the files it sees, as well as any subdirectories:

```
ls
  week1
```

You probably don't have any files in your home directory, so all we see is the subdirectory we created with the visual interface.

Since **week1** is a directory, we can use the **ls** command to take a peek inside it:

```
ls week1
  hello.cpp
```

and sure enough, the file you copied earlier shows up.

DIRECTORIES: Making Directories

We already created the **week1** directory with the visual interface.

We can make subdirectories with the command line interface as well. Let's create a folder for next week's work now, using the **mkdir** command:

```
mkdir week2
```

Now the **ls** command will display two directories:

```
ls
  week1  week2
```

DIRECTORIES: Moving Down to a New Directory

The **cd** command (*change directory*) is used to move from one directory to another in the command line interface.

If we are moving *down*, that is, into a subdirectory of the current directory, we just type the (short) name of our destination:

```
pwd
  /panfs/panasas1/users/jburkardt
ls
  week1 week2
cd week1
pwd
  /panfs/panasas1/users/jburkardt/week1
ls
  hello.cpp
```

DIRECTORIES: Moving Up to a New Directory

A directory can only have one directory "above" it. The abbreviation **..** (two dots) indicates this directory.

Let's take a journey up from week1 to home, down to week2, back up again to home, and finally back to week1:

```
pwd
  /panfs/panasas1/users/jburkardt/week1
cd ..
pwd
  /panfs/panasas1/users/jburkardt
cd week2
pwd
  /panfs/panasas1/users/jburkardt/week2
cd ..
cd week1
pwd
  /panfs/panasas1/users/jburkardt/week1
```

Introduction to Scientific Computing with C++

COMPILE: Finally, Let's Compile!

One reason for the command line interface is that it's a simple way to compile and run programs.

The compiler for C++ on our system is called **g++**.
Its full name is the Gnu C++ compiler.

The short and simple command to compile **hello.cpp** is:

```
g++ hello.cpp
```

If no errors occurred, we have a new file in our directory, the executable program **a.out**. We can run this by the command:

```
./a.out
  Hello, world!
```

COMPILE: Renaming a Program

You can change the name of a program (or any file) with the **mv** command.

Every time we use the compiler to make an executable program, it will be called **a.out** by default. So it might make sense to issue the following command:

```
mv a.out hello
```

This way, if we compile another program, we won't lose the "Hello, world!" program. Moreover, the name of the program reminds us of what it does. To run the renamed program, we would say:

```
./hello
  Hello, world!
```

COMPILE: Saving Program Output

Most programs print some kind of message to the user's screen. In C++, output to the user's screen is called *standard output*.

Sometimes, it is useful store program output in a file. You might want to mail it to someone, print it out, or save it for reference.

This is easy to do, using the **output redirection operator**, which is the "greater than" sign >.

If we issue the **ls** command, for instance, we can save the results by

```
ls > my_files.txt
```

and if we have renamed our "Hello, world!" program, we can save its output by

```
./hello > hello.txt
```

COMPILE: Saving Program Output

You can also save program output using the visual interface.

1. Run the program as usual.
2. Use the mouse to select the output on the screen.
3. Under the **Edit** menu on the **Terminal**, select **Copy**.
4. Start an editor (such as **kedit** or **gedit** and under the **Edit** menu, select **Paste**.

```
./hello
  Hello, world!
gedit hello.txt
```

(Now cut and paste the output on the screen into the empty file. Then save and close the file.)

Introduction to Scientific Computing with C++

PROGRAM: Edit a Program

It is too soon to expect you to create a C++ program on your own. However, we can go through the motions, by entering the text of a program that has already been written.

It's possible you have used an IDE (Interactive Development Environment) for doing programming, which is a more visual way to work on code.

But to start with, we will look at the simplest technique, using the same kind of text editor you would use to write a letter.

PROGRAM: Edit a Program

You can start up an editor in the Terminal:

- Type **pwd** to make sure you are in the **week1** directory.
- Type **kedit add_ints.cpp** to start the editor;
- Type in the lines on the next page, then save and exit.

or, using the Visual Interface:

- Choose **Applications / Accessories / Text editor**
- A blank window opens up; type in the lines on the next page.
- Choose the editor Menu item **File/Save**;
- Name the file **add_ints.cpp**, and use **Browse for other folders** to save it in the **week1** directory.
- Exit the editor

PROGRAM: Edit a Program

```
# include <iostream>
using namespace std;

int main ( )
{
  int number1, number2, number3;

  cout << "Enter first integer: ";
  cin >> number1;
  cout << "Enter second integer: ";
  cin >> number2;

  number3 = number1 + number2;

  cout << "The sum is " << number3 << "\n";
}
```

PROGRAM: Compile and Run Your Program

Using the terminal application in the **week1** directory, compile your program:

```
g++ add_ints.cpp
```

If any errors occurred, you may have to go back into the editor and try to correct them.

Now run your program, with the following input:

```
./a.out
  Enter first integer:  123456789
  Enter second integer: 987654321
  The sum is _______
```

This is the end of the in-class exercise. To get credit, please show Detelina your computer screen with the results, or save the output and email it to her at **dks10d@fsu.edu**

Introduction to Scientific Computing with C++

CONCLUSION:

When you work on the lab computers, you need to become familiar with the visual interface and the terminal interface.

We have learned some basic Unix commands today:

- **cd** to change directories
- **ls** to list files;
- **mv** to rename a file;
- **pwd** to report the present working directory;

and how to start some programs from the command line:

- **./a.out**, a user program with the default name of **a.out**
- **g++**, the compiler
- **kedit**, the editor;