Announcements

- HW1 is posted - due next Sat (individual this time)
- Lab tomorrow (pre-lab due before class tomorrow)
Command line tips - google UNIX tutorial/tips

In general you'll use 5-6 commands

- ls - list
- cp source$file target$file
- mkdir name
- rmdir name
- cd directory name
- mv source$file target$file
- rm file *careful!*
Others

- vi or emacs or pic

- g++

- man
Tricks

- Hitting the up arrow gives the last thing you typed (so then you can edit)
- Hitting tab will auto complete
- You can use & to get prompt back
  ex: `kak file &`

- `.` is current directory, `..` is parent
  ex: `cd ..`
  `cp ../*.file .`
Last time
- loops
- if
- functions

Today: input/output
Common error
What is wrong?

double gpa;
cout << "Enter your gpa: ";
cin >> gpa;
if (gpa == 4.0)
   cout << "Wow!" "endl;
Do-while loops

```cpp
int number;

do {
    cout << "Enter a number from 1 to 10: ";
    cin >> number;
} while (number < 1 || number > 10);
```

- Executes body before checking the boolean

```cpp
while ( ) { }
```
The main function

Every program defaults to running a

```c
int main() {
    body;
    return 0;
}
```
Arrays

Python has lists, tuples, etc.

In C++, only have arrays.

- Size is fixed at declaration
- Type is fixed (4 homogeneous)

Ex:

```c
int numbers[5];
numbers[0] = 55;
numbers[4] = 10;
```

```
numbers[5] = 5;
```

might work, might segfault
Creating Arrays:

**Allowed:**

```java
int daysInMonth = [31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31, 30];
```

**Error:**

```java
int daysInMonth [];
```

*must specify a size*

**Allowed:**

```java
char greeting[] = "Hello";
```
Multidimensional arrays

```c
for (int i = 0; i < 8; i++)
    for (int j = 0; j < 10; j++)
        table[i][j] = i + j;
```

**Table:**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>5</td>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
#include <iostream>

## Input + Output

C++ has several predefined classes.

<table>
<thead>
<tr>
<th>Class</th>
<th>Purpose</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>istream</td>
<td>Parent class for all input streams</td>
<td>&lt;iostream&gt;</td>
</tr>
<tr>
<td>ostream</td>
<td>Parent class for all output streams</td>
<td>&lt;iostream&gt;</td>
</tr>
<tr>
<td>iostream</td>
<td>Parent class for streams that can process input and output</td>
<td>&lt;iostream&gt;</td>
</tr>
<tr>
<td>ifstream</td>
<td>Input file stream</td>
<td>&lt;fstream&gt;</td>
</tr>
<tr>
<td>ofstream</td>
<td>Output file stream</td>
<td>&lt;fstream&gt;</td>
</tr>
<tr>
<td>fstream</td>
<td>Input/output file stream</td>
<td>&lt;fstream&gt;</td>
</tr>
<tr>
<td>istringstream</td>
<td>String stream for input</td>
<td>&lt;sstream&gt;</td>
</tr>
<tr>
<td>ostringstream</td>
<td>String stream for output</td>
<td>&lt;sstream&gt;</td>
</tr>
<tr>
<td>stringstream</td>
<td>String stream for input and output</td>
<td>&lt;sstream&gt;</td>
</tr>
</tbody>
</table>

- 5 new versions
Using iostream

#include <iostream>
using namespace std; X std::cin

Notes:
- can now use cin (for input)
  + cout (for output)
- separate distinct variables by \
  >> or <<
- use endl for end of a line
  "using namespace std" is (sort of) optional
Example

**Python**

```python
print "Hello"
print "Hello,"  # blank line
print first, last  # automatic space
print total
print str(total) + "."  # no space
print "Wait...",  # space; no newline
print "Done"
```

**C++**

```cpp
1 cout << "Hello" << endl;
2 cout << endl;  // blank line
3 cout << "Hello, " << first << endl;
4 cout << first << " " << last << endl;
5 cout << total << endl;
6 cout << total << "." << endl;
7 cout << "Wait... ";  // no newline
8 cout << "Done" << endl;
```
Formatting output

```cpp
cout << team << ": ranked " << rank << " of " << total << " teams" << endl;
```

- No `"%d` here to easily format

Can set precision:

```cpp
cout << "pi is " << fixed << setprecision(3) << pi << endl;
```

- Note that precision stays set to 3
Using `cin`

```cpp
int number;
cout << "Enter a number:"; cin >> number;
```

**Note:** - inputs are separated by any white space
  - `cin >> a >> b`
  - type of input must match type of variable (not all strings)
One possible problem:

```cpp
string person;
cout << "What is your name?"; cin >> person;
cin >> age; // I type "Erin Chambers".
What happens?

person = "Erin"; //
**GetLine**

- `getline` is a function which saves the string up to (but not including) the next newline.

**Ex:**
```cpp
String person;
cout << "What is your name?";
getline(cin, person);
```
Another tricky example

```c++
int age;
string food;
cout << "How old are you? ";
cin >> age;
cout << "What would you like to eat? ";
getline(cin, food);
```

I type: 15 hot dogs

Problem: age = 15
food = "h" or " "
Using File Streams - fstream

```cpp
#include <fstream>
using namespace std;

if file is known:
    int score;
    ifstream mydata("scores.txt");
    mydata >> score;

if not:
    ifstream mydata;
    string filename;
    cout << "What file? ";
    cin >> filename;
    mydata.open(filename.c_str()); // parameter to open must be a C-style string
```
ofstream
By default, writing to a file overwrites the file. (Think `"w"` in Python.)

To append:

```cpp
ofstream datastream("scores.txt", ios::app);
```
Reading and writing

There is also an `ifstream` object which allows reading and writing to a single file.

Much more complex.
String Streams

Ex: Converting between numbers & strings.

```
int age(42);
string displayedAge;
stringstream ss;
ss << age;
ss >> displayedAge;
```
A note on variable scopes:

```cpp
int main ()
{
    int a;
    if (a > 0)
        int b = 12;
    else
        int b = 16;
    cout << "a is " << a << endl;
    cout << "b is " << b << endl;
}
```