Announcement

(on board)
Command line tips - google "unix command line tutorial"

In general, you'll use 5-6 commands the most:

- ls
- cp sourcefile targetfile
- mkdir name
- rmdir name
- cd directory name
- mv sourcefile targetfile
- rm - Careful!
2 men 15

. men - manual pages

. 9++

. 1st or 2nd or Pic/phone

Others
ex: cd ..

- Is current directory, is parent

ex: cd / the gap prompt back

- You can use .. to get prompt back

- Hitting tab will autocomplete (a then you can edit)

- Hit enter to start arrow plus the

Tricks
\[ \angle C = x + 5 = 49^\circ \]

\[ \angle C = x + 2 = 25^\circ \]

What is wrong?

Common error
Remember: `countdown` function from Lab 3.0.

Defining a function: `example`
for (int count = start; count < end; count++)
    cout << count << endl;
}

void countdown(int start, int end)
{
    if (start > end) return;
    for (int count = start; count >= end; count--)
        cout << count << endl;
}

Optional arg is ignored.
If statements

if (bool) {  
  body 1;
} else if {  
  body 2;
}

Ex:

if (x < 0)  
  x = -x;

if (groceries.length() > 15)  
  cout << "Go to the grocery store" << endl;
else if (groceries.contains("milk"))  
  cout << "Go to the convenience store" << endl;

Note:  
- Don't need brackets if 1 line  
- don't need else  
- no else if
So nothing can get ugly.

If (1)

else

If (2)

else

Else

Else

3

{"code":
  "
  
  
  
  
  
  
  
  "}

{"code":
  "
  
  
  
  "}
anything not 0 or 1 is false

0 = false

Ex: if (mustBeCount < count)

which are really bools

written with numeric conditions can be

if/while bools

If/while statements can be
- Testcases body before checking the boolean

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

Do-while loops
3

```
3
return 0;
```

```
body:
```

```
int main()
{
    no input variables
```

```
Even if there are default values, a main function is needed to run the
```

```
The main function
```
Count all numbers [2] e.g.: [10]
numbers [4] = 10
numbers [5] = 55
numbers [5] = 0

Ex: int number [55];

Type is fixed (not homogeneous)
Size is fixed at declaration

In C++, only have arrays.
Python has lists, tuples, etc.
Careful of spelling!
Allowed: \( \text{char } \text{Greengr}
\begin{array}{c}
= \text{"Hello"}.
\end{array}
\)

One Exception

Enroll: \( \text{int days in Month} \)?

\begin{array}{c}
\end{array}

Can enroll: Allowed.

Created Arrays:
for (int i = 0; i < 8; i++)
for (int j = 0; j < 8; j++)
int table[8][8];

Multi-dimensional arrays
<table>
<thead>
<tr>
<th>Class</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>istream</td>
<td>Parent class for all input streams</td>
</tr>
<tr>
<td>ostream</td>
<td>Parent class for all output streams</td>
</tr>
<tr>
<td>iostream</td>
<td>Parent class for streams that can process input and output</td>
</tr>
<tr>
<td>stringstream</td>
<td>String stream for input and output</td>
</tr>
<tr>
<td>cout &lt;&lt; stream</td>
<td>String stream for output</td>
</tr>
<tr>
<td>cin &lt;&lt; stream</td>
<td>String stream for input</td>
</tr>
</tbody>
</table>

C++ has several predefined classes.
- Using namespace std is (Don't use)

- Use cerr for end of line

- Separate disjoint variables by

```
# include <iostream>
```

Notes: - Can now use cin (for input)
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: " << endl;
cin >> number;
```

**Note:** Inputs are separated by any white space.

```cpp
cin >> a >> b >> c
```

- type of input must match type of variable (not all strings)
`person = "Bin"

What happens if I type "Bin"? "Chomp".

Can "person" be a string? "What's your name?"

Is it possible to pass a problem?
Get line (can, person)?

Count. "What is your name?"

Shiny person.

Ex: next n.n. line The things up to (but not including) the next n.n. line is a function which succeeds
Problem: 3 + 5 = 8

I type: 15
mydata.open(filename.c_str());

if (!filename).

cout << filename;

if (strlen(filename))

isstream mydata;

if (!not):

isstream mydata("scores.txt");

if (!f)

using namespace std;

#include <fstream>

Using File Streams - Fstream
To append:

```python
(Think of it in Python.)
```

By default, writing to a file overwrites any previous contents.
Much more complex.

There is also an inherent object

Reading and writing
Ex: Casting between numbers & strings.
```c
3

count >> 6 >> a >> end;

int b = 16;

else
    int b = 12;

if (a < 0)
    int a;

int main ()
{

    A note on variable scopes;
```