The Edit-Compile-Run Cycle

EECS 211

Winter 2018

So you've written a program:

```
#include <iostream>

int main()
{
    std::cout << "Hello, world\n";
}
```

What now?

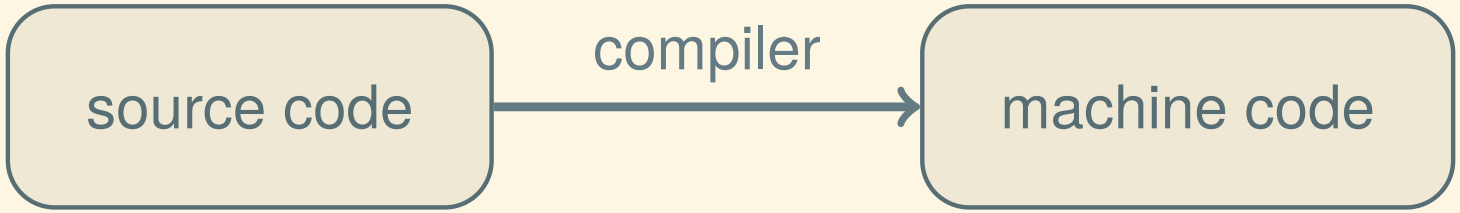
Compilation

We need to translate our program from

- source code (*e.g.,* C++, human readable)

to

- machine code (machine executable).



Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
$
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
$ cat hello.cpp
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
$ cat hello.cpp
#include <iostream>

int main()
{
    std::cout << "Hello, world\n";
}
$
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
$ cat hello.cpp
#include <iostream>

int main()
{
    std::cout << "Hello, world\n";
}
$ c++ hello.cpp -o hello
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
$ cat hello.cpp
#include <iostream>

int main()
{
    std::cout << "Hello, world\n";
}
$ c++ hello.cpp -o hello
$
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
$ cat hello.cpp
#include <iostream>

int main()
{
    std::cout << "Hello, world\n";
}
$ c++ hello.cpp -o hello
$ ./hello
```

Compilation in the shell

```
$ exec scl enable devtoolset-4 tcsh
$ mkdir eecs211
$ cd eecs211
$ emacs -nw hello.cpp
$ cat hello.cpp
#include <iostream>

int main()
{
    std::cout << "Hello, world\n";
}
$ c++ hello.cpp -o hello
$ ./hello
Hello, world!
$
```

Build management

As programs get larger, builds get more complicated:

- More files to compile, in complex combinations
- Want to just recompile the changed files
- Different compilers/machines want different options and work differently

Build management

As programs get larger, builds get more complicated:

- More files to compile, in complex combinations
- Want to just recompile the changed files
- Different compilers/machines want different options and work differently

We'll use a software building system called CMake to do all this for us

Getting a CMake project onto EECS

You can download an example CMake project from the course website:

```
$ cd eecs211
$ wget http://users.eecs.northwestern.edu/~jesse/
          course/eecs211/lec/01compile.zip
⋮
```

(No line break in the wget command—that's one long URL.)

```
$ unzip 01compile.zip
⋮
$ cd 01compile
$ ls
CMakeLists.txt  hello.cpp
$
```

Setting up CMake

You should have a project directory with a CMakeLists.txt file in it, like from the previous slide. Change to that directory. Then once, to set up CMake:

```
$ cd eecs211/01compile
$ mkdir build
$ cd build
$ ~jesse/pub/bin/cmake ..
-- The CXX compiler identification is GNU 5.3.1
⋮
-- Configuring done
-- Generating done
-- Build files have been written to:
   /home/jesse/eecs211/01compile/build
$
```

Building with CMake

Change directories to the CMake build directory. Use the make command to build, then run your program.

```
$ cd eecs211/01compile/build
$ make hello
Scanning dependencies of target hello
[ 50%] Building CXX object CMakeFiles/hello.dir/
                                hello.cpp.o
[100%] Linking CXX executable hello
[100%] Built target hello
$ ./hello
Hello, world.
$
```

Building with CMake

Change directories to the CMake build directory. Use the make command to build, then run your program.

```
$ cd eecs211/01compile/build
$ make hello
Scanning dependencies of target hello
[ 50%] Building CXX object CMakeFiles/hello.dir/
                                    hello.cpp.o
[100%] Linking CXX executable hello
[100%] Built target hello
$ ./hello
Hello, world.
$
```

After editing `hello.cpp` you can use `make hello` to build again.