Compiling Code Bases

Generating an Executable File

C File (.c)

Object File (.o)

 Intermediate machinespecific representation of just what is in a C file

Executable (no extension)

Compiler: translate from human readable to machine-specific code

Linker: bring together multiple object files so that all functions are known

Gnu C Compiler (gcc)

- Performs the compiling and linking phases for us
- Also invokes the assembler as part of the compiling process

Compiling Code Bases

As the set of files in a program gets large, we want to:

- Have a way to invoke the compiler easily
- Only compile the code that needs to be compiled
- Have a way to track which files depend on which other files

Invoking gcc at the compiler gets tiring and error prone...

Make Files

One of several ways to manage the compiling/project management process

- Define dependencies: what files depend on other files?
- Define rules for how to create derived files
 - Including the invocation of the compiler
- Uses file time stamps to know what work actually needs to be done

Our First Program

```
#include <stdio.h>
int main(int argc, char** argv)
{
   printf("Hello, World\n");
}
```

gcc hello.c -o hello

Our First Makefile

```
# The top rule is executed by default all: hello
```

Other rules are invoked as necessary

Rule for creating the hello executable hello: hello.c

gcc hello.c -o hello