

MAKE ME A SANDWICH.

|

SUDO MAKE ME
A SANDWICH.

|



WHAT? MAKE
IT YOURSELF.

/

OKAY.

/



Our Virtual Machine

Ubuntu 16.04

- Install VirtualBox
- Download VM image from course web site
- Import image
- Boot image

(written details coming)

(Some) Unix Commands You Should Know

- pwd
- ls
- ls -l
- ls -la
- touch
- cat, more, less
- mkdir, rmdir
- rm
- rm -rf
- cd
- hexdump
- wc
- top
- make

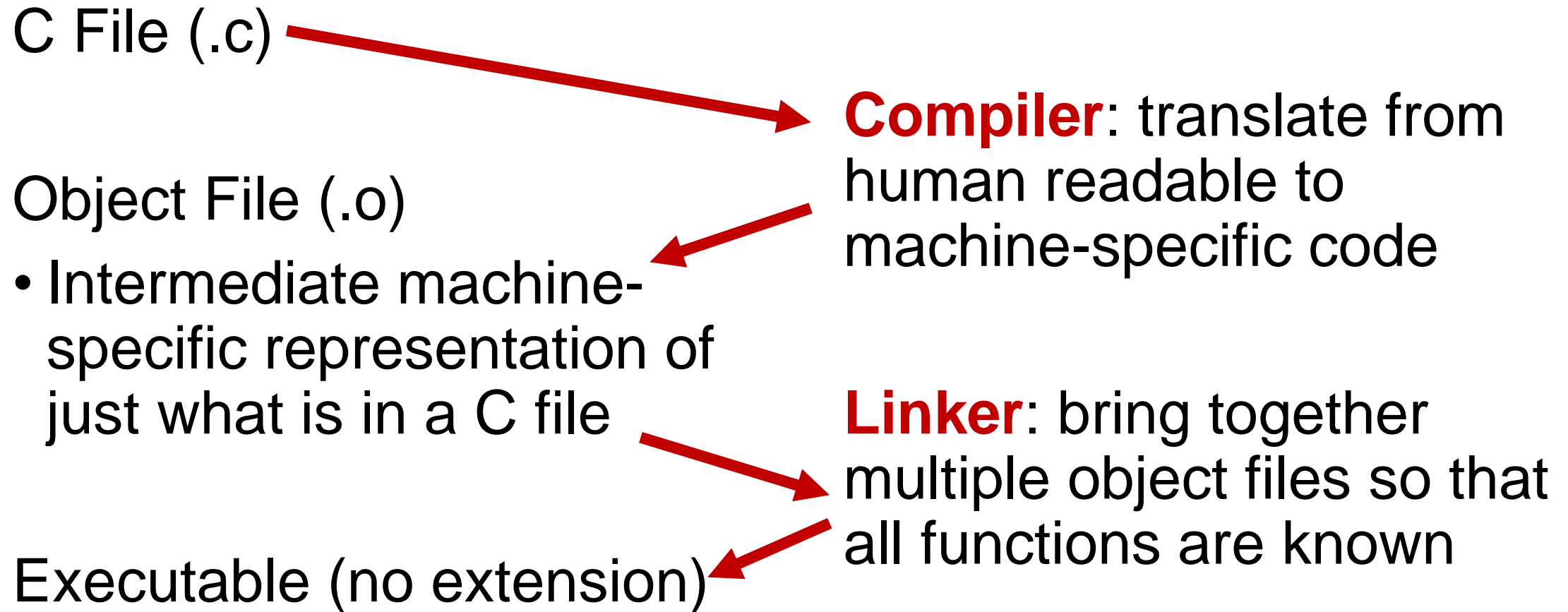
Other Key Unix Concepts / Tools

- Absolute vs relative paths
- Editors
 - emacs
 - nano
 - vim
- sudo
 - Execute commands as the administrator
 - `sudo shutdown -h now`

Compiling Code Bases

- Gnu
- Not
- Unix

Generating an Executable File



Gnu C Compiler (gcc)

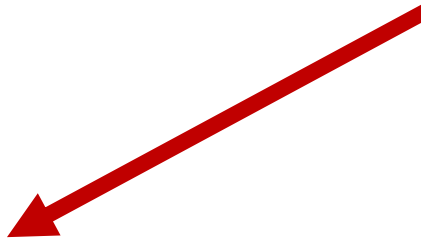
- Performs the compiling and linking phases for us
- Also invokes the assembler as part of the compiling process

Executing an Executable File

Executable (no extension)



Loader: bring together
executable and
dynamically linked libraries



Executing 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
```

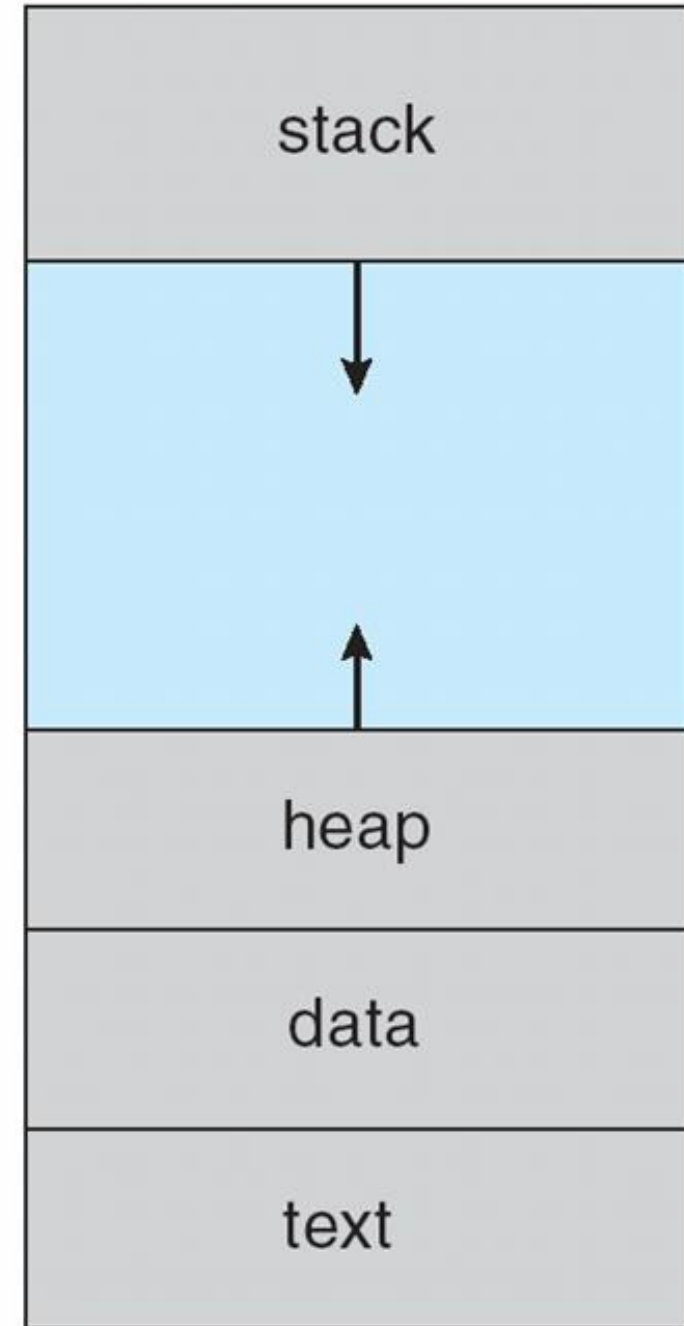
Live demo...

Processes and Memory

On process creation, the process is effectively given its own memory space

- Text: storage of code
- Data: global variables (preallocated space)
- Heap: dynamically allocated space
- Stack: local variable storage

max



Stack and Heap

- Stack grows downward with each nested function call
 - Local variables, register state, return memory address
- Heap
 - Storage of dynamically allocated items that must be persistent across function calls (and returns from function calls)
 - OOP languages: object instantiation is done in the heap

Variables

- Every variable declaration results in an allocation of memory
- For primitive data types (int, char, float), the name of the variable refers to the value that is stored in the corresponding memory location
- However, we can get at the actual memory location

```
int a;
```

```
a = 5;
```

```
&a refers to the address in memory
```