Unix CLI and Some C

Goals Today...

- Getting connected to the Linux machines.
- Principles of Unix Shell commands.
- Command line text editing.
- Makefiles and compiling C programs.
- Compiling a Linux Kernel.

Connecting to Linux Machines

- There are four Linux boxes in O/U 308
  - tron.cs.kzoo.edu
  - pong.cs.kzoo.edu
  - joust.cs.kzoo.edu
  - frogger.cs.kzoo.edu
- The only way to connect remotely is through ssh.
  - From a Mac, start a terminal and type:
    - ssh yourusername@curie.cs.kzoo.edu
  - From a PC, download putty, and follow the instructions.

Why Bother with a CLI?

- Convenient for low bandwidth connections.
- There are always some jobs that are difficult or impossible with a GUI.

Unix Commands

- Many too many to talk about here.
- A few useful ones.
  - `pwd, ls, cp, mv, rm, mkdir`
  - `cat, less, more, tail`
  - `grep, who, diff, wc, date`
  - `man`

Providing Input to Commands

- Commands generally take two types of input:
  - Flags change the way the command works,
    - `ls`
    - `ls -l`
    - `ls -la`
  - Files specify the data the the commands will act on.
    - `ls -l tmp.txt`
    - `cp tmp.txt tmp_cpy.txt`

Specifying Files

- Files may be specified according to relative or absolute paths.
  - Absolute:
    - `ls -l /home/nsprague/myfile.txt`
  - Relative
    - `ls -l myfile.txt`
    - `ls -l ../myfile.txt` (“`..`” indicates directory above the current directory)

Wildcards

- Most commands that accept file input, accept wildcards that allow pattern matching.
- * represents anything.
- ? represents any single character.
- `cat *.txt`
  - prints the contents of every file that ends in .txt.
- `cat file?.txt`
  - prints file1.txt, file2.txt, filea.txt, etc.

Redirection

- Output can be redirected to a file:
  - `ls -l > file_list.txt` (creates)
  - `ls -l >> file_list.txt` (appends)
- Input can be read from a file:
  - `grep thestring < look_in_file.txt`

Pipes

- Output from one command can be “piped” into the input of another:
  - who | ls -l
  - who | grep sprague | wc -l > num_sprague.txt

Building Programs in C

- Unlike Java, C programs typically have separate files for declarations (.h) and definitions (.c).
- If one file_a.c needs to access methods defined in file_b.c, then it will include a statement like:
  - `#include “file_a.h”`
- This is a preprocessor directive.
- It does exactly what you would expect.
- C compilation really has three steps:
  - preprocess -> compile -> link.

Sample C Program

- fibonacci.h:

Protects against repeated includes.

```
#ifndef FUNCTIONH_DEFINED
#define FUNCTIONH_DEFINED

int fibonacci(int n);

#endif
```

- fibonacci.c:

```
#include "fibonacci.h"

int fibonacci(int n) {
  if (n <=1) {
    return 1;
  } else {
    return fibonacci(n-1) + fibonacci(n-2);
  }
}
```

Continued

- main.c

```
int main(int argc, char* argv[]) {

  int num = atoi(argv[1]);
  printf("%d\n", fibonacci(num));
}
```

- argc and argv???
- Command line arguments:
  - argc is the number, argv is an array of char*'s (strings)

Pointers to Functions

- Let's look at some sample code...

Compiling C Programs: The Simple Way

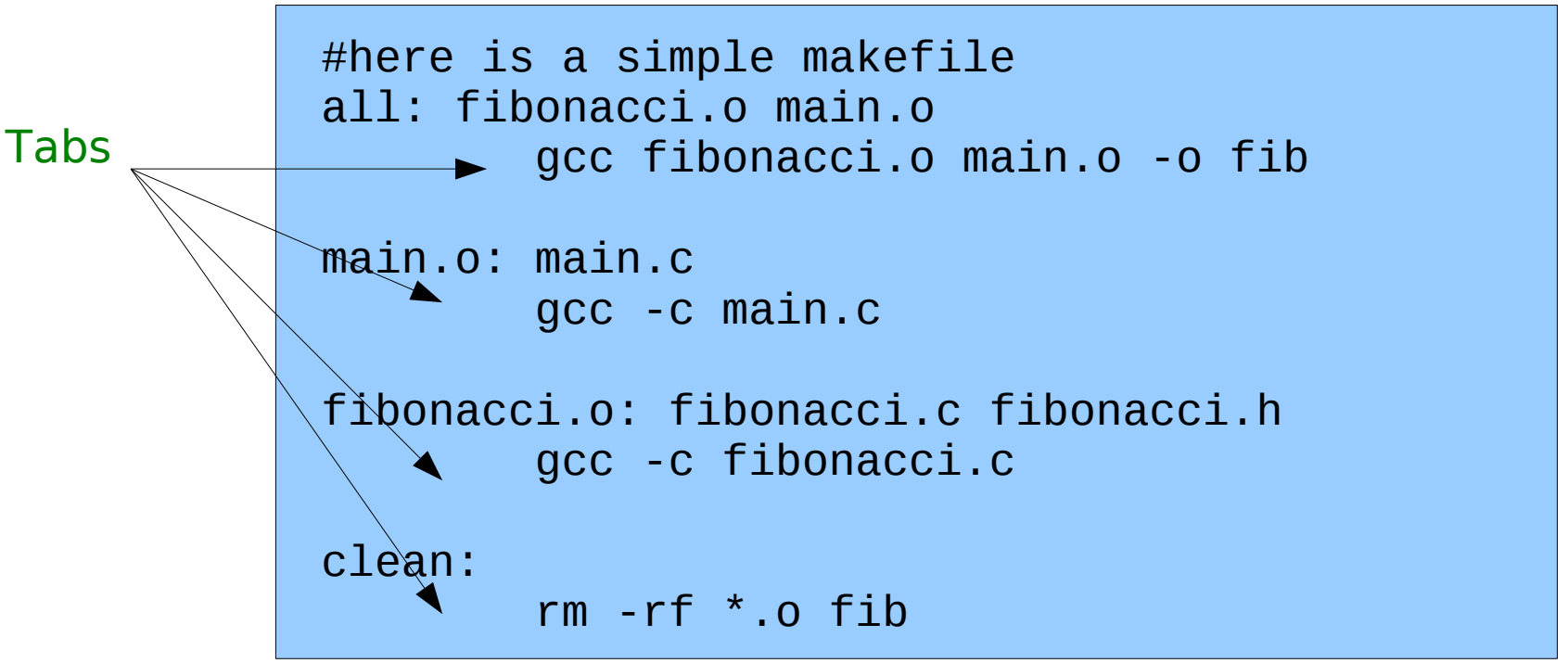
- gcc the_code1.c the_code2.c
  - Results in an executable named a.out.
  - Note that we don't need to list any .h files.
- gcc the_code1.c the_code2.c -o the_program
  - Results in an executable named the_program.

Compiling C Programs

- What if you have 33432 .c files, and you only changed one?
- We can create object files (.o) that can be linked into the final executable.
  - .c -> compiler -> .o -> linker -> executable
- We can recompile just one file, then relink.
- Sounds like a pain to keep track of...
  - make is a utility for keeping track of dependencies when building large programs.
  - Gets its input from makefiles.

Makefiles

- Simple makefile:



- Format is:

```
target: dependencies
<TAB>action
```

More on Makefiles

- It is also possible to use variables:

```
#here is a simple makefile
CC = gcc
CFLAGS = -c -g
all: fibonacci.o main.o
	$(CC) fibonacci.o main.o -o fib

main.o: main.c
	$(CC) $(CFLAGS) main.c

fibonacci.o: fibonacci.c fibonacci.h
	$(CC) $(CFLAGS) fibonacci.c

clean:
	rm -rf *.o fib
```

- There is much much more...

Debugging C Programs

- The GNU debugger is gdb.
- Notice the -g on the previous slide.
- Tells gcc to include debugging information in the executable.
- Running the debugger:
  - `gdb your_executable`
- Setting a breakpoint:
  - `break fibonacci.c:6`
  - `run`

Compiling and Installing A Linux Kernel

- Let's do it...

Linux Coding Preview

- Even when compressed, the Linux kernel takes up about 45 megabytes.
- Problem is keeping track of many different versions related by small changes.
- Solution is diff/patch.
- Creating a patch:

diff -uprN linux-2.6.12-vanilla linux-2.6.12-withchanges > /tmp/patch

- Applying a patch (in the top level source directory.)

```
patch -p1 < /tmp/patch
```