Slide 1

Administrivia

- Homeworks 1, 2, and 3 graded. Homework 4 will be graded soon. Sample solutions on Web.
- Homework 6 (LaTeX) and 7 (makefiles) to be on Web soon.
- In the syllabus I said you would each be required to do a project of some sort. But maybe homeworks would be easier/better. So, two options:
  - Homeworks only. 100 points for homework, 50 points for attendance.
  - Homeworks and project. 70 points for homework, 30 points for project, 50 points for attendance.

  Which should we do? *(The decision was to choose the former, with an optional extra-credit project.)*

Slide 2

The `make` Utility

- Motivation: Most programming languages allow you to compile programs in pieces (“separate compilation”). This makes sense when working on a large program — when you change something, just recompile the parts that are affected.
- Idea behind `make` — have the computer figure out what needs to be recompiled and issue the right commands to recompile it.

Slide 3

Makefiles

- First step in using `make` is to set up a “makefile” describing how the files that make up your program (source, object, executable, etc.) depend on each other and how to update the ones that are generated from others. Normally call this file `Makefile` or `makefile`.

  Very simple example:

```
main:   main.o foo.o
        gcc -o main main.o foo.o
main.o: main.c defs.h foo.h
        gcc -c main.c
foo.o:  foo.c
        gcc -c foo.c
```

- When you type `make`, `make` figures out (based on files’ timestamps) which files need to be recreated and how to recreate them.

Slide 4

Useful Command-Line Options

- `make` without parameters makes the first “target” in the makefile. `make foo` makes `foo`.
- `make -n` just tells you what commands would be executed — a sort of “dry run”.
- `make -f otherfile` uses `otherfile` as the makefile.

Slide 5

Makefiles, Continued

- You define dependencies by giving, for each "target", a list of files it depends on.

  You also give the list of commands to be used to recreate the target.

  NOTE!: Lines containing commands must start with a tab character.

- You can also define "phony targets". Example:

```
clean:
        -rm *.o main
```

Slide 6

Makefiles, Continued

- You can also define variables, e.g.:
  - A list of object files needed to create an executable. Then use this list in the list of dependencies, the command, etc.
  - The pathname for a command, options to be used for all compiles, etc.
- Example:

```
objs = main.o foo.o
CFLAGS = -Wall -pedantic
main:   $(objs)
        gcc $(CFLAGS) -o main $(objs)
```

Slide 7

Predefined Implicit Rules

- `make` already knows how to “make” some things — e.g., `foo` or `foo.o` from `foo.c`.
- In applying these rules, it makes use of some variables, which you can override.
- A simple but useful makefile might just contain:

```
CFLAGS = -Wall -pedantic -O
```

- Or you could use

```
CFLAGS = -Wall -pedantic $(OPT)
OPT = -O
```

and then optionally override the `-O` by saying, e.g., `make OPT=-g foo`.

Slide 8

Implicit Rules (Pattern Rules)

- You can define similar rules — e.g., a makefile to compile `.cpp` files using the MPI C++ compiler:

```
MPICXX = /usr/bin/mpiCC
CCFLAGS = -O -Wall -pedantic
LDFLAGS = -llammpi++

%: %.c
        $(MPICXX) -o $@ $(CCFLAGS) $< $(LDFLAGS)
```

`$<` is the `.c` file here (first prerequisite), and `$@` is the target.

(Note that this is for GNU `make`. Non-GNU `make` has a similar idea — “suffix rules” — with slightly different syntax.)

Other Uses For `make`

Slide 9

- `make` can be used to automate things other than compiling programs. It's particularly useful for defining implicit rules.

  Example: Makefiles to run `latex` and associated programs.

Minute Essay

Slide 10

- How are we doing so far? Are you getting what you hoped to get out of this course? Is the workload about right?
- We've now covered all the basics. Things I had in mind to talk about include
  - Useful/standard Unix utilities (e.g., `tar`).
  - Text-based mail programs and/or mail filtering with `procmail`.
  - CGI scripts.

  Or I'm open to other suggestions! (though topics about which I know nothing might take more prep time than I have).
- Reminder: Homework 5 due today.