

Slide 2

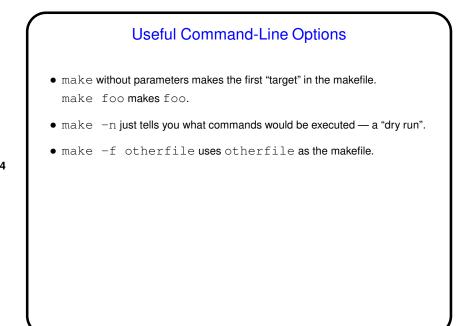
1

Makefiles

• First step in using make is to set up "makefile" describing how 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.

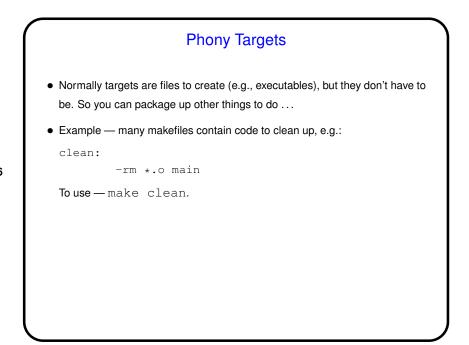
Slide 3

- Simple example on "sample programs" page.
- When you type make, make figures out (based on files' timestamps) which files need to be recreated and how to recreate them.



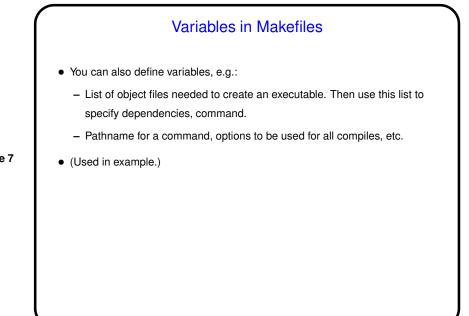
Slide 4

Defining Rules
Define dependencies for a rule by giving, for each "target", list of files it depends on.
Also give the list of commands to be used to recreate target. *NOTEI:* Lines containing commands must start with a tab character. Alleged paraphrase from an article by Brian Kernighan on the origins of UNIX: The tab in makefile was one of my worst decisions, but I just wanted to do something quickly. By the time I wanted to change it, twelve (12) people were already using it, and I didn't want to disrupt so many people.

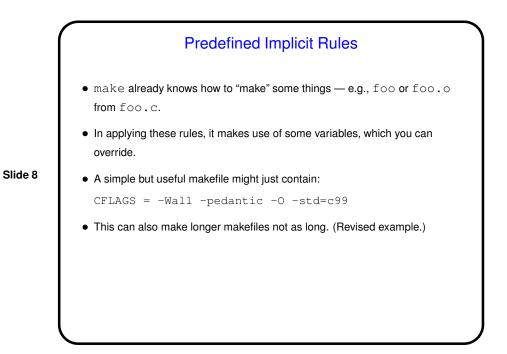


Slide 6

3



Slide 7



make — Overriding Variables at Runtime

• Something else that can be useful in makefiles is providing variables that can be overridden at runtime. For example, if in the makefile you have

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

```
Slide 9
```

you can override SOPT with e.g., make $\operatorname{OPT}=-g$ foo.

Minute Essay • Have you used make in other courses? (I seem to remember hearing that Dr. Fogarty uses it in Functional maybe?) Slide 10

5