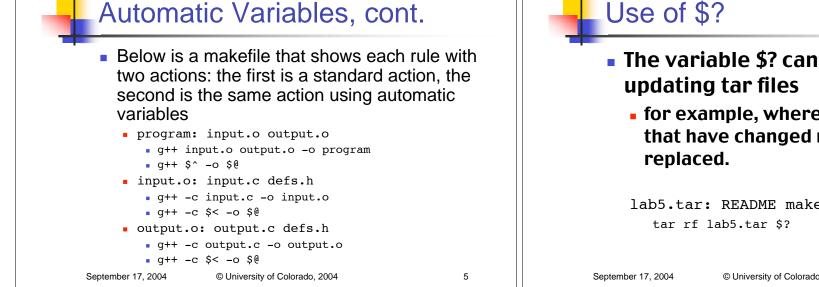


3

September 17, 2004

4



Use of \$?

- The variable \$? can be useful for
 - for example, where only those files that have changed need to be

lab5.tar: README makefile lab5.cpp

© University of Colorado, 2004

6

The View Path

- Make applies special meaning to another variable, VPATH, also know as the view path.
 - While VPATH is not an automatic variable, it does interact with them (as we shall see shortly)
- VPATH consists of a list of directories, just like the path variable of the shell
 - If make cannot find a dependency in the current directory, it looks in the view path.
 - Note: just because a file is found in the view path does not mean that the shell can find it when executing commands (see next slide)

View Path Example

VPATH = \$(HOME)/csci3308/src/lab05 lab05.o: lab05.c

q++ -c lab05.c -o lab05.o

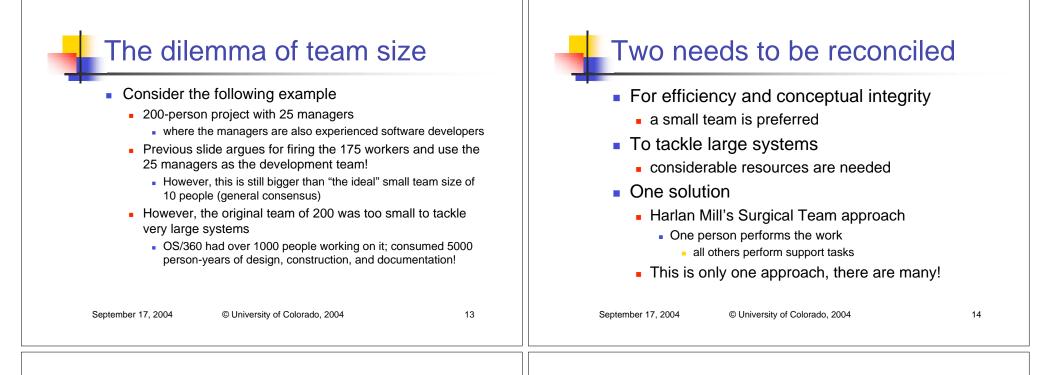
% make

q++ -c lab05.c -o lab05.o

g++: lab05.c: No such file or directory

- Why does the action fail?
 - Assume we invoked the command in a build directory

<section-header><list-item></list-item></section-header>	 View Path Example, continued VPATH = \$(HOME)/csci3308/src/lab05 lab05.0: lab05.c g++ -c \$< -0 \$@ make g++ -c /home//src/lab05/lab05.c -0 lab05.o Here, the action references lab05.c correctly. Note: lab05.o is created in the current directory, even though the source code is located elsewhere (which is similar to how we compiled gnuchess in lab 1)
September 17, 2004 © University of Colorado, 2004 9	September 17, 2004 © University of Colorado, 2004 10
Accessing File Information	Brooks' Corner: The Surgical Team (Chapter 3)
 Using automatic variables, a file's name and directory can be extracted <pre>VPATH = \$(HOME)/csci3308/src/lab05 lab05.o: lab05.c echo found file \$(<f) \$(<d)="" %="" directory="" echo="" file="" found="" home="" in="" lab05.c="" lab05<="" make="" pre="" src=""></f)></pre>	 Or How should a development team be arranged? The problem Good programmers are much better than poor programmers typically 10 times better in productivity typically 5 times better in terms of program elegance but we often do not have access to these "super programmers"



The Proposed Team

- The surgeon
 - The chief programmer
- The co-pilot
 - Like the surgeon but less experienced
- The administrator
 - Relieves the surgeon of administrative tasks
- The editor
 - Proof-edits documentation

- Two secretaries
 - Support admin and editor
- The program clerk
 - Probably obsolete today
- The toolsmith
 - Supports the work of the surgeon
- The tester
- The language lawyer

How is this different?

- Normally, work is divided equally
 - Now, only surgeon and copilot divide the work
- Normally, each person has equal say
 - Now, the surgeon is the absolute authority
- Note communication paths are reduced
 - Normally 10 people => 45 paths
 - Surgical Team => at most 13 (See Fig. 3-1.)

September 17, 2004

How does this scale?

- Reconsider the 200 person team
 - Communication paths => 19,900!
- Create 20, ten-person surgical teams
- Now, only 20 surgeons must work together
 - 20 people => 190 paths
 - Two orders of magnitude less!
- Key problem is ensuring conceptual integrity of the design

The Modern Surgical Team

- The surgical team, as conceived by Mills and described by Brooks, is not widely used today
- On Internet time, the chief programmer approach is impractical
- Now, it is more important that there be one to three designers, or software architects, that guide the design of the system
 - with many people implementing the system
 - This is true of the Microsoft approach
 - The program manager is responsible for the feature set

18

It is also true of many open source programs September 17, 2004 © University of Colorado, 2004

September 17, 2004 © University of Colorado, 2004 17