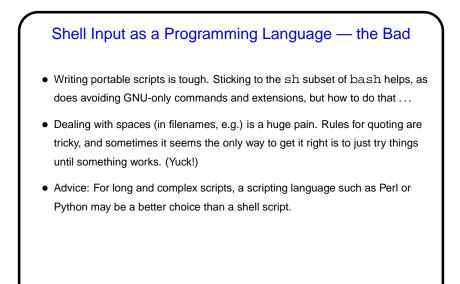
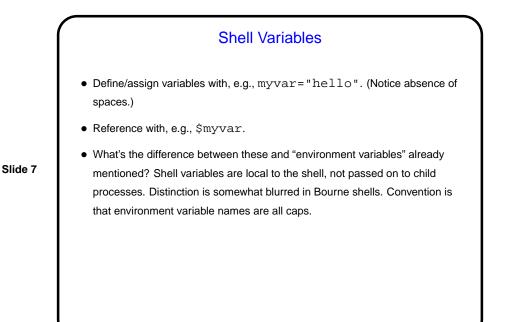


Shell Input as a Programming Language — the Good
What bash understands is in a sense a programming language, with the shell as its interpreter:
Variables (usually untyped).
Expressions (arithmetic and logical).
Conditionals (if/then/else) and loops.
Functions.
Can be used interactively, or collected into "scripts".
I will talk about bash, but most shells provide similar functionality, just sometimes with different syntax.

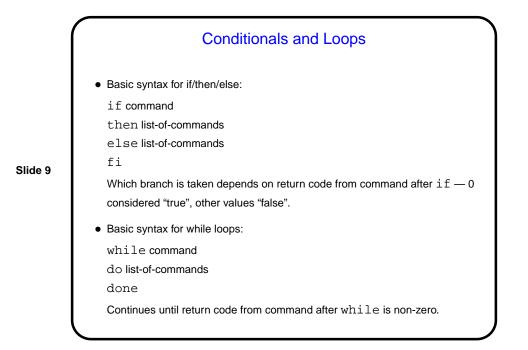


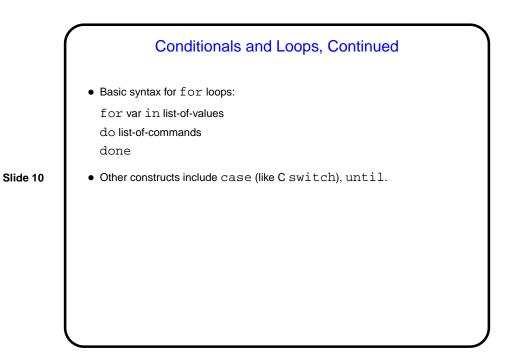
Shell Scripts
A "shell script" is just a sequence of things you could type at the shell prompt, collected in a (text) file.
Normally, first line of script is # ! followed by path for shell (/bin/bash, e.g.), and the file is marked "executable" (with chmod). But you can also execute commands in file anyfile via bash anyfile.
With the exception of the first line, lines starting with # are comments.

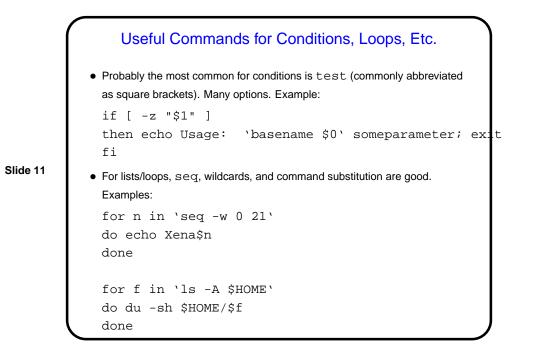
Slide 5



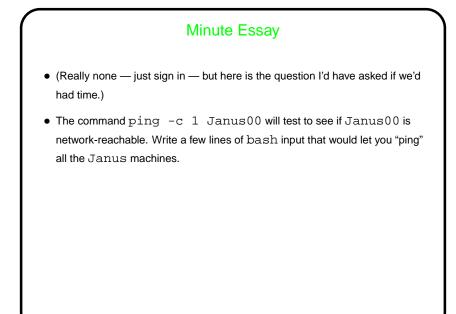
Shell Functions and Parameters
Define functions as described previously — function followed by name, parentheses, then function definition in curly brackets. Separate/end commands with *i* or newlines.
Parameters for functions and shell scripts are positional — \$0 for function name, then \$1, etc. \$* is a list of all parameters; \$# is the count of parameters, not including \$0.
Call functions or shell scripts by giving name and then parameters, separated by whitespace. (If a parameter should include whitespace, use quoting or escape characters.)







Other Features • Evaluating (numeric) expressions — next time. • Reading from standard input — next time. Slide 12



	Minute Essay Answer
Slide 14	 One possible answer: for n in `seq -w 0 21` do ping -c 1 Janus\$n done Another answer (contributed by a student one year):
	for n in `ruptime grep Janus awk '{print \$1}' do ping -c 1 Janus\$n done