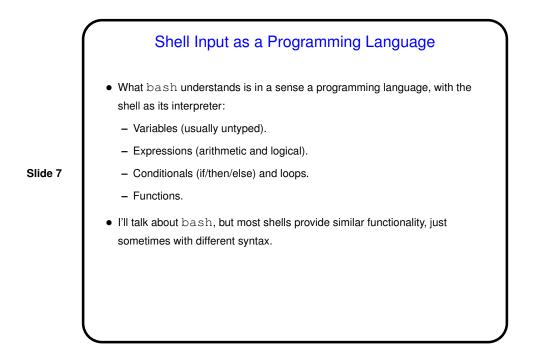
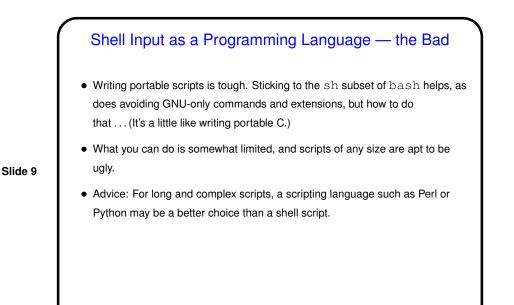


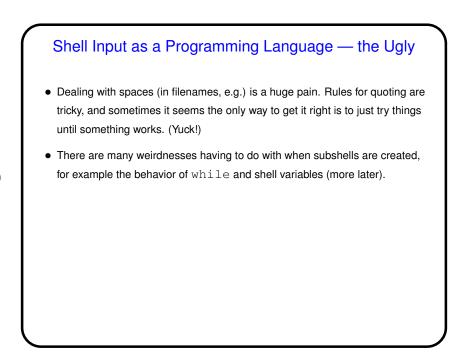
basename and dirname split up pathname into "base" (last level of path) and rest of path.
Very helpful in combination with command substitution, especially in scripts.

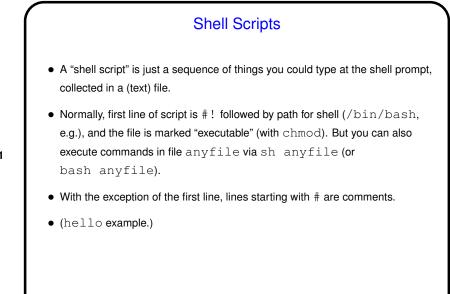


Shell Input as a Programming Language — the Good

- Interactive shells are a kind of REPL (read, evaluate, print loop) for the shell's language. So you can use the various features interactively or use them to write "scripts" in the same way you can test out ideas in Scala's REPL and then use them in programs (except that Scala's REPL is mostly useful for testing/development, whereas using shell features such as loops interactively can be useful).
- Any UNIX/Linux system will have a shell of some sort, I think always one that supports basic sh functionality, while which "real" programming languages are available might vary.

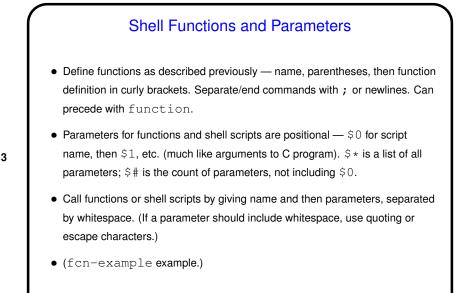


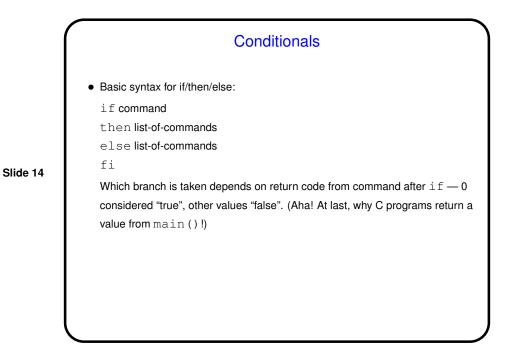


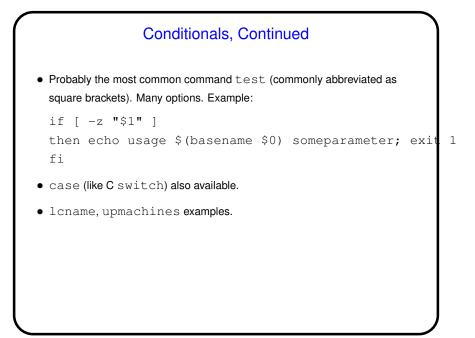


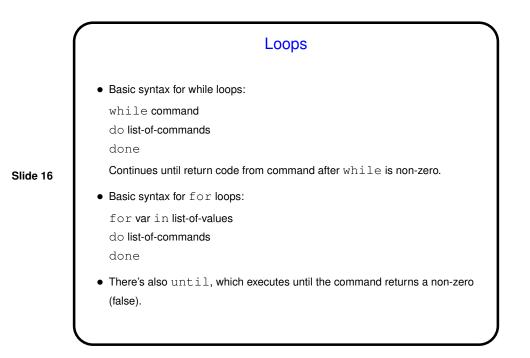
Slide 12

Shell Variables
Define/assign variables with, e.g., myvar="hello". (Notice absence of spaces.)
Reference with, e.g., \$myvar.
What's the difference between these and "environment variables" already mentioned? Shell variables are local to the shell, not passed on to child processes. Distinction is somewhat blurred in Bourne shells. Convention is that environment variable names are all caps.



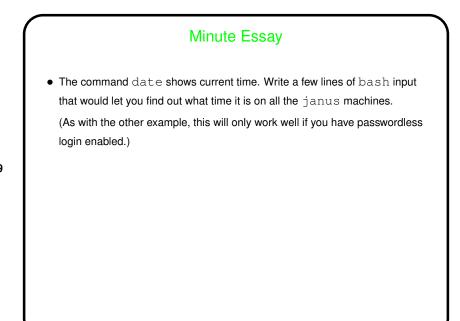






Loops — Examples
• A silly example (runs until interrupted):
while true
do
 date ; sleep 1
done
• Another somewhat silly example:
for n in `seq 0 5`
do
 ssh janus0\$n hostname
done
(Note that this only works well if you have your account set up to allow
passwordless login. You can find instructions for setting that up on my home
page.)





	Minute Essay Answer
Slide 20	 One possible answer: for n in 'seq -w 0 24' do ssh janus\$n date Another answer (contributed by a student one year): for n in 'ruptime grep janus awk ' {print \$1}' ` do ssh janus\$n date done