Commands and Arguments - Types of commands; argument splitting; writing scripts.

|  |
| --- |
| **Review:** • ***Arguments***: These are additional words specified after the command (’ls -l foo’ executes ls with two arguments: -l and foo). • ***Quotes***: The two forms of quotes, single and double (' and "), are used to group words and can protect special characters.  => Everything inside single quotes becomes a literal string  => *prevents some substitutions but allows others*.  - Every substitution that begins with a dollar sign $ is performed,  - as is the legacy `...` (backtick) [command substitution](http://mywiki.wooledge.org/CommandSubstitution).  - Backslash \ escaping is also performed.  - *No word splitting or filename expansion* is performed **Additionally:** • Tip — *Always* quote sentences or strings that belong together, even if it’s not absolutely necessary. This developed practice will reduce the risk of human error in the scripts. (e.g. quoting a sentence of an echo command). • FAQ — I’m trying to put a command in a variable, but the complex cases always fail! BashFAQ/0504. • FAQ — How can I handle command line arguments (options) to my script easily? BashFAQ/0355. |

# 1. I'm trying to put a command in a variable, but the complex cases always fail!

|  |
| --- |
| Variables hold data.  Functions hold code.   * Don't put code inside variables! There are many situations in which people try to shove commands, or command arguments, into variables and then run them. Each case needs to be handled separately.   *For the simple case in bash*, *you can use an array to store arguments to pass to a command* (similar to constructing a command using args only known at runtime, below):  args=(-s "$subject" --flag "arg with spaces")  mail "${args[@]}" |

## 1.1. Things that do not work

|  |
| --- |
| Some people attempt to do things like this:  # Example of BROKEN code, DON'T USE THIS.  args=$address1  if [[ $subject ]]; then  args+=" -s $subject"  fi  mail $args < "$body"  Adding quotes won't help, either:  # Example of BROKEN code, DON'T USE THIS.  args="$address1 $address2"  if [[ $subject ]]; then args+=" -s '$subject'"; fi  mail $args < "$body"  This fails because of [WordSplitting](http://mywiki.wooledge.org/WordSplitting) and because the single quotes inside the variable are literal, not syntactical   If $subject contains internal whitespace, it will be split at those points  The mail command will receive -s as one argument, then the first word of the subject (with a literal ' in front of it) as the next argument, and so on  Read [Arguments](http://mywiki.wooledge.org/Arguments) to get a better understanding of how the shell figures out what the arguments in your statement are.  Here's another thing that won't work:  # BROKEN code. Do not use!  redirs=">/dev/null 2>&1"  if ((debug)); then redirs=; fi  some command $redirs  Here's yet another thing that won't work:  # BROKEN code. Do not use!  runcmd() { if ((debug)); then echo "$@"; fi; "$@"; }  The runcmd function can only handle **simple commands** with no redirections. It can't handle redirections, pipelines, for/while loops, if statements, etc. |

## 1.2. I'm trying to save a command so I can run it later without having to repeat it each time

|  |
| --- |
| Just use a function:  pingMe() {  ping -q -c1 "$HOSTNAME"  }  [...]  if pingMe; then .. |