Shell Scripting with Bash

If, Then, Else

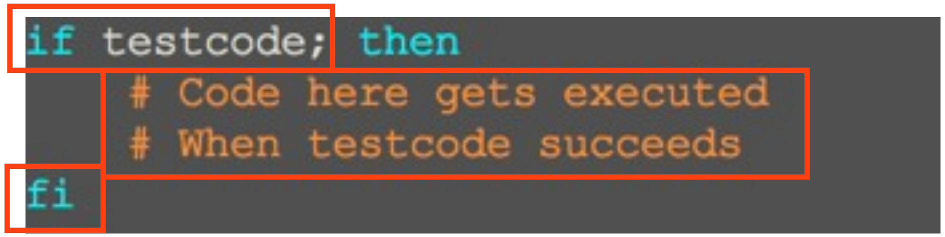
Reindert-Jan Ekker
http://nl.linkedin.com/in/rjekker/
@rjekker

pluralsight
hardcore developer training

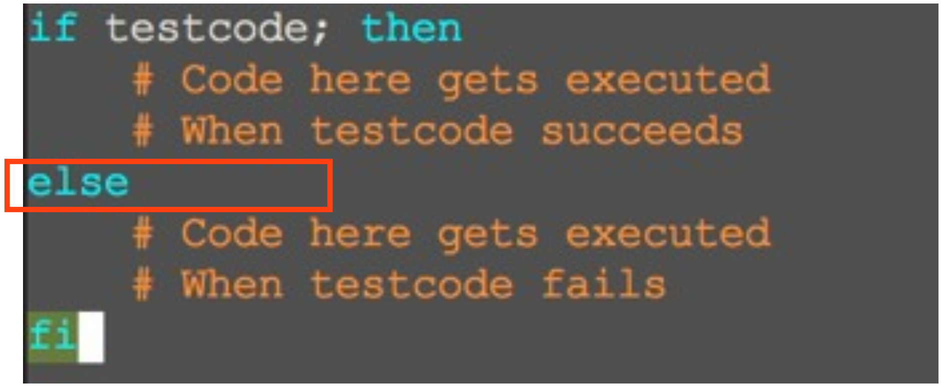
Overview

- If, then, else
- Return codes
- Conditional expression

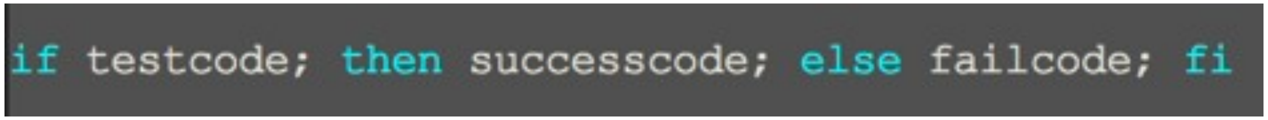
If



```
if testcode; then
    # Code here gets executed
    # When testcode succeeds
fi
```



```
if testcode; then
    # Code here gets executed
    # When testcode succeeds
else
    # Code here gets executed
    # When testcode fails
fi
```



```
if testcode; then successcode; else failcode; fi
```

- **Keywords if, then, else, fi**
  - First on a line, or
  - After a semicolon
- **help if**

Return codes

- **Return code or exit status:**
  - Value returned by program upon exit
  - 0..255
- **0 means success**
  - other values are error codes

- **Shell scripts return values with exit**
  - exit 0
- **Good habit: make sure your program exits with a correct value**
  - Always call exit with a value

- **If statement just looks at return code for "testcode"**

Conditional expressions

- **Conditional Expression**
  - Tests on files and directories
  - Tests on strings
  - Arithmetic tests

- **[[*Expression*]]**

Expression	True if
[[$str]]	str is not empty
[[$str = “something”]]	str equals string “something”
[[$str=”something”]]	**always returns true!**
[[-e $filename]]	file $filename exists
[[-d $dirname]]	$dirname is a directory

  - Spaces around the expression are very important!
  - Same for switches (-e) and equals sign

Conditional Expressions 2

- **Classical command: “test”**
  - Also: [
  - Harder to use, easier to make mistakes
  - Only use for portability
- **[[...]] is a bash extension**
  - Not a command but special syntax
  - No quotes needed around variables
  - Good habit: use [[..]] instead of [..]
- **Getting help**
  - “help test” will show you most important info
  - “help [[“ will tell you about the extension

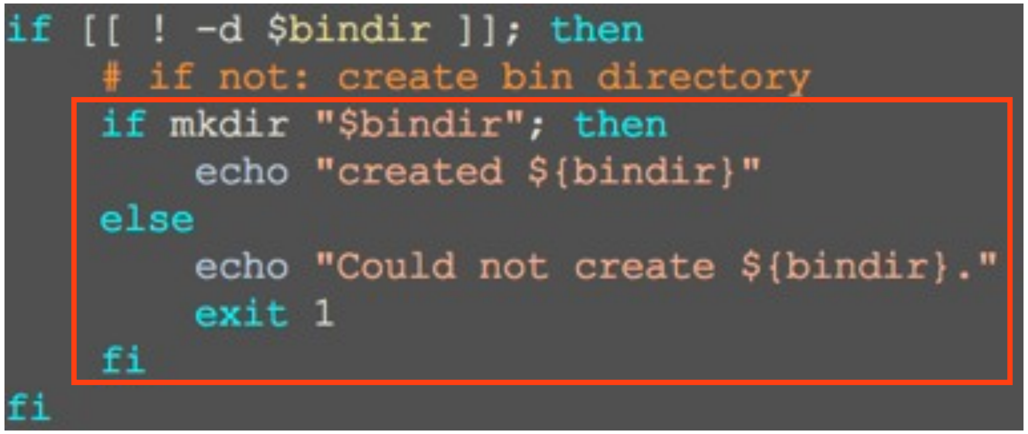
Arithmetic tests

- **For comparing integers only**
- **[[arg1 OP arg2]]**
- **Where OP is:**
  - -eq: equality
  - -ne: not equal
  - -lt: less than
  - -gt: greater than
  - And some others.. see help
  - So don’t use =, >, < for numbers!

- **Special variables:**
  - $# contains number of script arguments
  - $? contains exit status for last command
- **To get the length of the string in a variable:**
  - Use ${#var}

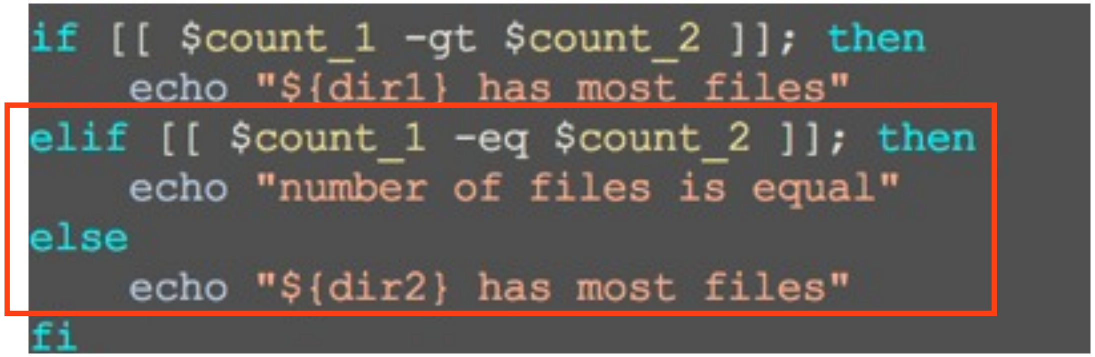
If again

- **Nested if**



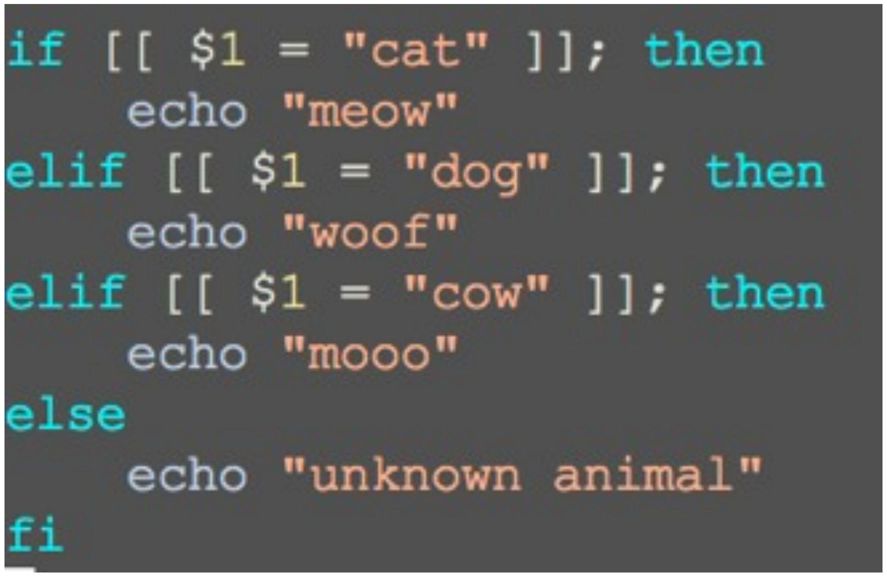
```
if [[ ! -d $bindir ]]; then
    # if not: create bin directory
    if mkdir "$bindir"; then
        echo "created ${bindir}"
    else
        echo "Could not create ${bindir}."
        exit 1
    fi
fi
```

- **Elif**



```
if [[ $count_1 -gt $count_2 ]]; then
    echo "${dir1} has most files"
elif [[ $count_1 -eq $count_2 ]]; then
    echo "number of files is equal"
else
    echo "${dir2} has most files"
fi
```

Multiple elifs



```
if [[ $1 = "cat" ]]; then
    echo "meow"
elif [[ $1 = "dog" ]]; then
    echo "woof"
elif [[ $1 = "cow" ]]; then
    echo "mooo"
else
    echo "unknown animal"
fi
```

- Each elif gets tried in turn
- If everything fails: run else

- This is like a switch statement in other languages

And, Or, Not

- **In a conditional expression:**
- **Use ! to negate a test:**
  - [[! -e $file]]
  - Use spaces around !

- **Use && for “and”:**
  - [[$# -eq 1 && $1 = “foo”]]
  - True if there is exactly 1 argument with value “foo”

- **Use || for “or”:**
  - [[$a || $b]]
  - True if a or b contains a value (or both)

- **Don’t use -a, -o for and, or**
  - Even though “help test” says so

Summary

- **If, then, else**
  - elif
  - Nested if
- **Return codes**
  - 0 = succes, everything else = failure
  - exit 0
- **Conditional expressions**
  - Use [[..]]
  - Don’t use [..] or test
  - “help test” and “help [[“
  - Testing strings, files, numbers
  - &&, ||, !