Shell Scripting with Bash

Shell Functions

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

Overview

- **Functions**
    - declare
    - use
    - return data
    - export

Functions

- **Define your own command**
- **name () { ... }**
  - You can run the code in the braces as a new command
  - other equivalent syntax (not recommended):
  - function name () { ... }
  - function name { ... }
- **Execute it like any command**
  - Give it arguments
  - Use redirection
- **Positional parameters are available for function arguments**
  - $1, $2, ...

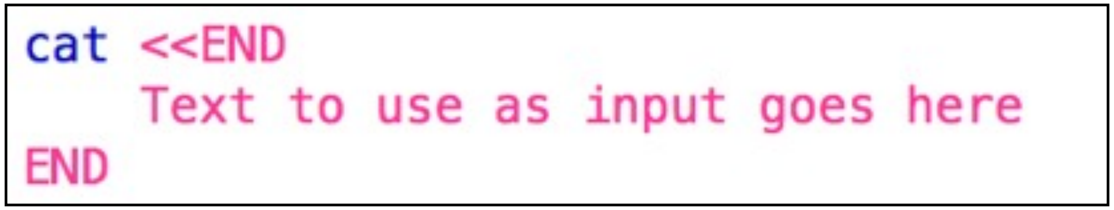
- **Naming your functions**
  - same rules as for naming scripts: don’t override existing commands

Functions 2

- **Bash variables are globally visible**
  - In a function, you can make a variable local to that function
  - Use declare or local
- **Exit a function with return**
  - returns a status code, like exit
  - Without a return statement, function returns status of last statement
- **Returning any other value**
  - Use a global variable
  - Or send the data to output and use command substitution
- **Exporting a function**
  - export -f fun

Miscellaneous

- **Functions and redirection**
  - Redirection is allowed immediately after function definition
  - Will be executed when function is run
  - fun () { ... } >&2
- **A command in a pipeline runs in a subshell**
  - ls | while read -r; do ((++count)); done
  - du -d 0 */ | read_filesizes
- **Here documents:**
  - Have a command read its input from the source file
  - << Tag
  - Tag defines end of input



```
cat <<END
    Text to use as input goes here
END
```

Summary

- **fun () {...}**
- **Calling a function**
  - positional parameters
  - redirection
- **Return data**
  - return value
  - output
  - global variable
- **Here documents**
- **A command in a pipeline runs in a subshell**