Shell Scripting with Bash

Control Flow

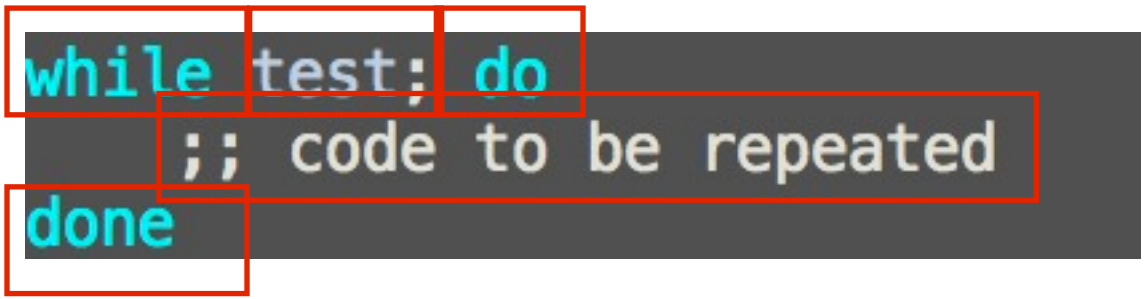
Reindert-Jan Ekker
http://nl.linkedin.com/in/rjekker/
@rjekker

pluralsight
hardcore developer training

Overview

- **Loops**
  - while/until
  - for
- **Break and continue**
- **Case**
- **Compound commands**
- **|| and &&**

while



- Repeats code in the block
- Continues as long as test returns true

until

```
until test; do
    ;; code to be repeated
done
```

- **Repeats code in the block**
- **Continues as long as test returns false**

The classic for loop

```
for VAR in WORDS; do
    ;; code to be repeated
done
```

- Assign each word in WORDS to var in turn
- Will stop when no words are left
- Do NOT quote WORDS

The C-Style for loop

```
for (( INIT; TEST; UPDATE )); do
    ;; loop code
done
```

- Use double parentheses
- First expression: initialize your loop variable(s)
- Second expression: a test. The loop will run as long as this is true
- Third expression: update the loop variable(s)

- Expressions use syntax for arithmetic evaluation
  - (this is explained later)

Break and Continue

- **break**
  - quits the loop
- **continue**
  - skips the rest of the current iteration
  - continues with the next iteration

- **Both can be used in for, while and until**

case

```
case WORD in
    PATTERN1)
        code for pattern 1;;
    PATTERN2)
        code for pattern 2;;
    ...
    PATTERNn)
        code for pattern n;;
esac
```

- **Matches word with patterns**
  - Pattern matching is the same as with matching filename patterns
  - Use *, ?, []
- **Code for first pattern that matches gets executed**
- **End code with ;;**
  - so you can use multiple statements separated by ;
- **Multiple patterns separated by |**

Command Groups

- **Group commands with {}**
  - Will group them into a single statement
  - Can use I/O redirection for the whole group
  - Use the group in an if statement or while loop
  - Return status is that of the last command in the group
- **{ cmd1; cmd2; cmd3; }**
  - Separate the commands with newlines or semicolons
  - Use spaces around braces
  - Ending semicolon or newline not optional!

|| and &&

- **Execute next statement depending on return status of previous statement**
  - Basically: short for if
- **&&**
  - Will execute next statement only if previous one succeeded
  - mkdir newdir && cd newdir
- **||**
  - Will execute next statement only if previous one failed
  - [[$1]] || echo “missing argument” >&2

- **[[$1]] || echo “missing argument” >&2 && exit 1**
  - Dont do this: will always exit!
  - [[$1]] || { echo “missing argument” >&2; exit 1; }

Summary

- **Loops**
  - while/until
  - for
- **Break and continue**
- **Case**
- **Compound commands**
- **|| and &&**