Oscar Petersson, Matteus Laurent
oscpe262, matla782

TDDI41 Lab report

Högskoleingenjörsutbildning i datateknik, 180 hp

Laboration report - October 9, 2019
System Administration
TDDI41, Linköpings universitet

Assistant:
Jon Dybeck
IDA

SCT

Exercise 1: $* and $@

1-1 What is the difference between using $* and $@ to access all parameters to a script or function?

`$@` looks at each argument individually, whereas `@*` regards them as a collection as shown by the example below:

```
#foo.sh
#!/bin/bash
printf "%s\n" "$@"
printf "%s\n" "$*"
#EOF

$ ./foo.sh meep moop
  meep
  moop
  meep moop
```

Exercise 2: Positional parameters in for loops

2-1 Why is $@ quoted (inside quotation marks) in the for loop?

`$@`

will make another evaluation of the parameter which can yield unwanted results with reserved characters. For instance, if fed `‘ls -F‘` when an executable is present in the path (annotated in the output with `’*’`, e.g. `foo*`), it will evaluate the expression (`foo*` becomes *”all files beginning with*`"foo"`) which might result in duplicate arguments.

2-2 Could $* have been used instead of $@? Explain your answer.

Yes, and no. `$*` could work, but we would still run into the issues mentioned in *2-1* with evaluation of the arguments provided. `$*` would not work however, unless it is only one file provided as argument.

Exercise 3: While loops

3-1 How do while loops work.

```
#/bin/bash
i=10
while [ $i -ne 0 ]
do
    echo "$i"
    i=$(($i - 1))
done
```

Exercise 5: Bash syntax

5-1 What does ${line:0:1} do? Note that it is a form of parameter expansion.

Evaluates `$line`, string position 0, 1 character(s).

5-2 What does a colon on a single line do (the true branch of the if statement in the example).

: == true, which is needed as conditional blocks can't be empty.

5-3 Bash has an alternate syntax for command substitution (backticks). What is it?

```
$(command)
```

5-4 What does $((...)) do? What is this called in the bash man page.

Arithmetic expansion. Allows evaluation of arithmetic expressions and substitution of the result, allowing nesting for instance.

5-5 How could you count the number of dotfiles using a single pipeline (no loops or variables, just simple commands). You may want to read about commands like grep and wc.

```
$ ls -A | grep "^\." | wc -l
```

Exercise 6: More bash syntax

6-1 Look at the first two lines of the script (after the initial "#!" line).

a) What does the backslash at the end of the first of these mean.

A backslash is used as en escape character in bash, and in this case it escapes the newline, allowing the script to be written on several (two here) lines while being evaluated as if it was a single line.

b) What do the braces around echo and exit mean? What would happen if you used parenthesis instead of braces.

Earlier, we saw that command scope within () are executed in a new sub-shell. If replaced by {} the scope is executed in the current shell context.

6-2 What does the local command do at the beginning of smtp_send.

`local` creates a temporary object that is local to the current call frame, in this case a variable `$mailserver`.

6-3 Why call sleep 1 in the script.

To demonstrate `sleep`?

7-1

7-1 is implemented as SCT7.sh (with much of the work being done in functions found in common.sh) in the test suite.

8-1

The full test suite is run through tddi41.sh.