Using Shell Variables

Reindert-Jan Ekker

@rjekker www.codesensei.nl

Overview

Variables

- Create
- Assign value
- Use value

Pass parameters to our script

Best practices

- Naming
- Quoting

Variables

```
# To create a variable: just assign a value

# Don’t use whitespace around =

container=A3

filename=shipments.csv

# Values containing spaces: use quotes

filenames=”notes.txt picture.jpg movie.mov”

# To retrieve the value, prefix with $

echo $x
```

Shell variables have no data type. They simply store strings.

Naming

- Use letters, numbers, underscore
- First character: a letter or underscore
- Case-sensitive

Uppercase variables

- pre-defined variables
- PATH, HOME, SECONDS, IFS, etc.
- Don’t override them by mistake

Good Habit

- Use lowercase names

Quote Your Variables

Use "$x" instead of $x

No surprises when input contains spaces

Use double quotes

- keep meaning of dollar sign intact

Not necessary in zsh

- Zsh does not split variables in words
- Behaviour is configurable

Braces

Where does your variable name end?

`echo "${foo}bar"`

- prints value of var `foo`
- followed by string "`bar`"

`echo "$foobar"`

- prints value of `foobar`

Taking Arguments

```
# Your script can accept input from arguments
$ create_report.sh A5 myfolder


# Inside the script, the arguments are available
# As special numbered variables
container="$1"    # The first argument
directory="$2"    # Second argument
third_arg="$3"    # etc.
```

End of Options

```
# If your input might start with a -
# Use -- (end of options)
grep -- “$user_input”
rm -- “$file_to_delete”

# Some commands don’t support this
# Use printf instead of echo
printf “I will delete the file: %s\n” “$file_to_delete”
```

```
#!/bin/bash -x

# This does not work with /usr/bin/env, use set instead

#!/usr/bin/env zsh
set -x
```

Debugging

Pass option to the interpreter (works with bash and zsh)

-v prints each line before running it

-x prints each line including variable values

Debugging

Show each line with variable values as the script runs

Regular shebang

```
#!/bin/bash -x
```

Or

```
#!/bin/zsh -x
```

With /usr/bin/env

```
#!/usr/bin/env zsh
set -x
```

https://www.shellcheck.net/

Awesome utility that
detects bugs and mistakes

Export

Variables are local to your script

- Or terminal session

Export a variable

- To make it available to subprocesses
- `export var`
- `export var="value"`

Up Next:
Conditional Execution