Using the shell

Johan Öhlin

11 November 2021

Yabs

Debugging files

Files for exercieses should be available at:
`https://github.com/Najoj/yabsbashcourse/tree/main/exercises/5`

Note: The notes will tell you which file to try, but you are free to use whichever you want. Therefore, only `script.bash` will be used for the examples.

Debugging

Print every line executed:

```
$ bash -x script.bash
```

or, set `-x` argument in script head

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

or, set with `set` command inside script

```
#!/bin/bash
set -x # Note the minus sign
# ...
set +x # Note the plus sign
```

Debugging

Print shell input lines as they are read:

```
$ bash -v script.bash
```

or, set `-v` argument in script head

```
#!/bin/bash -v
```

or, set with `set` command inside script

```
#!/bin/bash
set -v # Note the minus sign
# ...
set +v # Note the plus sign
```

Can of course be combined with `-x`.

Debugging

Download `script1.bash` from Github without and with commands above.

Note what happens and how they differ.

Is the output expected?

Debugging

Show full execution path with `-ilx`.

Tip: Long output can be stored in a file by using

```
$ bash -ilx script.bash &> save.txt
```

Or read through less `$ bash -ilx script.bash 2>&1 | less`

The execution steps are written to stderr. `2>&1` will redirect stderr to stdout. `less` will only read streams from stdin and only data from stout will be directed to stdin.

Debugging

Using debug variable `DEBUG`.

```
$ DEBUG bash script.bash
```

```
#!/bin/bash
if ${DEBUG+x}; then
  # Set debug options an variables
fi
```

Note: If parameter is null or unset, nothing is substituted, otherwise the expansion of word is substituted.