

Shell script

June 4

Shell script

- sh: A programming language specification
- bash, csh, zsh: implementation of sh
 - With their own extensions
- Shell script: a script that is written in sh
 - Run using interpreter
- To execute 'a.sh', either run as 'bash a.sh', or './a.sh'
 - In the latter case, the file should have permission 'executable' (chmod +x a.sh)

Checking a shell script

- shellcheck: brew/apt install spellcheck
 - Online version: <https://www.shellcheck.net/>

```
In ./mybuild.sh line 9:
```

```
-DZ3_INCLUDE_DIR=$Z3DIR/include \
```

```
^----^ SC2086: Double quote to prevent globbing and word splitting.
```

- VSCode: Bash IDE, shellcheck

She-bang (#!)

- “#!/bin/bash” at the first line of a script
- “#!”: format indicator
 - Unix reads first a few bytes to identify the format of the file
 - Indicates that this is a script (not ELF executable file)
- The remaining characters indicate the interpreter to use
 - This can be /usr/bin/python , etc
- Note that # is also used as a character to begin comment

Variables

```
# Define a variable X
X="hi"
# Use the variable with prefix $
echo $X
```

- Variables can be defined from the outer space: “environment variable”
 - A user executes ``export X=hi``
 - Execute ``a.sh`` (which does ``echo $X``)
- It cannot be exported to the outer space though
 - ... because it is run as a child process
 - To allow it to export values, run ``source a.sh``

Execution of commands

```
cp bitcount1.ll input.ll
timeout 60 sf-compiler input.ll -o a.s
# Check whether $? is 124
```

- A shell script contains a list of commands
- Execution of a command finishes with a return code: the integer from its `int main()`
- If it is zero, the command finished successfully
- Otherwise, 'something' happened:
 - `timeout`: returns 124 if timeout, otherwise the exit code of `sf-compiler`
- It will fall-through (does not exit immediately); to exit the script, use 'set -e'
 - To activate: use 'set +e'

If-then-else

- The space between tokens is important (marked those as underbars)
- -eq, -ne: integer compare
- ==, !=: string compare
 - Can use regular expression as well
- Prefer [[..]] over [..] *
- More operators: <https://www.tldp.org/LDP/abs/html/comparison-ops.html>

```
timeout 60 sf-compiler a.ll ...
CODE=$?

if_ [[ _$CODE_ -eq 124_ ]]; then
    echo "Timeout"
elif [[ $CODE -ne 0 ]]; then
    echo "Failure"
else
    echo good
fi
```

* <https://stackoverflow.com/questions/669452/is-double-square-brackets-preferable-over-single-square-brackets-in-ba>

String

- No quote, Double-quote(""), single-quote("")
 - No quote: a string without any space (e.g. apple)
 - Double-quote: can use a variable inside a string (e.g. "hello world", "hello \$name")
 - Single-quote: cannot use a variable (e.g. 'hello \$name': name is not expanded)
- Backticks(``): execute the command and get the result from stdout!
 - FILELIST=`ls -l mydir`; echo \$FILELIST

For loop

```
for i in apple banana pear "black bean"; do  
    echo $i # prints fruits per a line  
done
```

- For loop iterates over items that are separated by spaces
- To iterate over a range, you can use `for i in {1..10}` that is a bash extension
 - Note that this does not allow `$N` to be used as lower/upper bound *
- Combined with ``ls -l`` or ``find . -name "*.txt"``, you can iterate files inside the dir
 - Should not have spaces in there names: use `find . -name "..." -exec`

* <https://stackoverflow.com/questions/169511/how-do-i-iterate-over-a-range-of-numbers-defined-by-variables-in-bash>