HPC.NRW

INTRODUCTION TO LINUX

(in an HPC context)

Version 20.09 | HPC.NRW Competence Network

THE COMPETENCE NETWORK FOR HIGH-PERFORMANCE COMPUTING IN NRW.

SHELL SCRIPTS

HPC.NRW Competence Network

INTRODUCTION TO LINUX

SHELL SCRIPTS

- Interaction with Linux: just a series of commands
  - Commands can be put into a text file
  - Text file is fed to console
  - Console runs commands one after the other

- Advantage: very easy automation

- Shell script: execute like a program
  - Remember “execute” permissions

EXECUTING SHELL SCRIPTS

- Command to run script
  - Full script name (including location)
  - Commonly: `./scriptname.sh`

- Why not only script name?
  - Linux only looks up commands in specific folders
    - Safety feature (not everyone can run everything)

- File needs execute permissions
  - Another safety feature
  - Remember `chmod` command (e.g. `chmod u+x`)

EXAMPLE SHELL SCRIPT

HPC.NRW

```
#!/bin/bash

# This is a comment line.
echo "Hello world."

ls -l
sleep 3s
ls \
 -l
```

So-called “shebang”

- Always has to be first line
- Comment plus exclamation point
- Specifies interpreter (here bash)
- Does not have to be Linux console (/usr/bin/python)

Comment symbol

- Line comments only
- Sometimes meta-commands

Echo command

- Common command
- Debugging, logging

List of commands

- Same as when entered manually

Line break

- Backslash as last character

VARIABLES

- Store output of commands
- Assignment via `=` (equal sign)
  - Example: `var="value"`
  - Important: no spaces around `=`
  - Always text
  - Quotes necessary when whitespace, special characters in value
- Retrieve with `$` sign

  `$var`
  - Example: `echo $var` prints value to screen

VARIABLES

- Common newbie trap: brackets and quotes in variables
  - Single quotes: exact text
  - Double quotes: variables will be expanded
  - Parentheses (round brackets): command inside will be evaluated

- `var=“bla”` will save the text bla to var
- `var=‘$bla’` will save the text $bla to var
- `var=“$bla”` will look for a variable named bla
- `var=$(bla)` will execute command bla and save its output to var

SHELL SCRIPTS: ADDITIONAL TIPS

HPC.NRW

- Use command line arguments: `$0` - `$9`, `${10}`
  - Example: script was called with `script.sh -f 5.0`
  - Then: `$0=script.sh`, `$1=-f`, `$2=5.0`
- Loops and if statements, similar to most programming languages

```
for file in $( ls ); do
    echo item: $file
done

if [ -e $filename ]; then
    echo "$filename exists."
fi
```

SHELL SCRIPTS: VARIOUS THINGS

- Shell scripts are good for running series of commands
  - Not so good for more complex programming
    - Loops, ifs etc. are an afterthought
    - I don’t know of an IDE or debugger
    - Can delete wrong file(s) very easily
  - Better: “proper” scripting language (e.g. Python)

- Default shell in most Linux systems (e.g. Ubuntu, CentOS): `bash`
  - Many alternatives: C-Shell(`csh`), Z Shell(`zsh`), Fish(`fish`)
    - Often completely different syntax
    - Prefer portable shell programming where possible