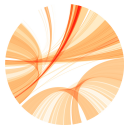
May 2019

BIO609: Unix

Part 2: Bash scripting

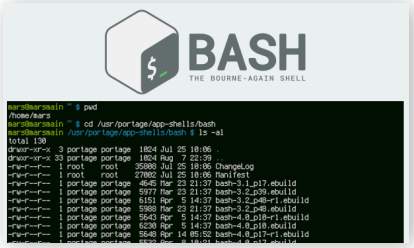
With short code snippets and exercises



Universität Zürich[UZH]

gregor_rot

What is bash?



Bash is a **Unix shell** and **command language**

- default login shell
- **command processor** that runs in a text window (**console**)
- user types commands, which are executed

Bash can also **read and execute commands from a file**

- **script file** also called a **shell script** (usually named .sh)
- **filename globbing** (wildcard matching, * ?)
- **piping** |
- $**variables**
- **control** structures, **condition** testing, **iteration** (if, for)

My first bash script

```
$ vi first_script.sh
```

vi is in command mode
press **i** (enters **insert mode**)

```
#!/bin/bash
echo "My first script :)"
```

press **ESC** (enters **command mode**)
type ":wq"

```
$ chmod +x first_script.sh
$ ./first_script.sh
```

set executable flat to first_script.sh
and run the script

```
bash script :)
```

output on the screen

Writing to a file

Special character **>** redirects output to a file

```
#!/bin/bash
#this is a comment
echo "Hello World" > log.txt
echo "How are you" > log.txt
```

→ writes "Hello World" to log.txt

→ overwrites log.txt with "How are you"

Special character **>>** appends output to a file

```
#!/bin/bash
#this is a comment
echo "Hello World" > log.txt
echo "How are you" >> log.txt
```

→ writes "Hello World" to log.txt

→ appends "How are you"

Variables

Variables are "named boxes" for values

- name=value (no spaces around =)
- to access, use $ sign
- enclose variable name in {} if using in combination with other names
- some environment variables that are defined by the shell, like $HOME

```
#!/bin/bash
home_folder=/home/gregor
echo "My home folder is $home_folder"
echo "My data folder is ${home_folder}/data"
```

Write a script that will store the environment variable $HOME to the file home_folder.txt

Arrays

Arrays are simply variables with multiple values (lists)

- name=(value1 value2 value3 ...)
- access by index starting from 0

```
#!/bin/bash
color=(red green blue)
echo ${color[*]} # echo all values from array
color[3]=yellow  # add new value to array
echo "The sky is ${color[2]}"
echo "There are ${#col[*]} colors in my array"
```

FOR loop

Iterate over values and repeat parts of code

```
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Current iteration is $i"
done

for i in {1..100}
do
    echo "Current iteration is $i"
done
```

Write a script that will print out all numbers between 1 and 1000 that are dividable by 13.

FOR loop (more examples)

Iterate over values and repeat parts of code

```
#!/bin/bash
for fname in /home/gregor/*.fasta
do
   echo "Current FASTA file $fname"
done

colors=(red green blue)
for color in ${colors[*]}
do
    echo "The color is $color"
done
```

Write a script that will put the numbers from 1 to 1000 that are dividable with 13 into an array and print the entire array at the end.

FOR loop, a convenient list of strings

```
#!/bin/bash

list="item1
item2
item3
item4"

for item_name in $list
do
   echo $item_name
done
```

WHILE loop

Iterate until conditions are met

```
#!/bin/bash
count=0
while [[ $count -lt 4 ]]
do
   echo "Current count is $count"
   let count+=1
done
```

IF statement

Check a condition and reach depending on the outcome

- can be composed of many **elif** statements
- always tested in order of appearance
- **else** clause is optional
- boolean operator to test if file exists: -e
- string1 = string2, string1 != string2

```
#!/bin/bash
if [[ something1 ]]
then
   command1
elif [[ something2 ]]
then
   command2
else
   command3
fi
```

```
#!/bin/bash
if [[ -e log.txt ]]
then
echo "log file exists"
else
echo "log
```

```
#!/bin/bash
name=Bob
if [[ $name= "Rod"]]
then
echo "Your name is Rod"
else
```

IF cheat sheet

FILES conditions

-e FILE	does FILE exist?

STRING conditions

string1 **=** string2	are strings equal?
string1 **!=** string2	are strings different?

INTEGER conditions

int1 **-eq** int2	are numbers equal?
int1 **-ne** int2	are numbers not equal?
int1 **-lt** int2	is int1 < int2 ?
int1 **-gt** int2	is int1 > int2 ?
int1 **-le** int2	is int1 <= int2 ?
int1 **-ge** int2	is int1 >= int2 ?

EXPRESSIONS

!exp	negates expression (logical NOT)
exp1 **&&** exp2	logical AND
exp1 **||** exp2	logical OR

IF another example

```
#!/bin/bash
for i in {1..4}
do
  if [[ $i = 1 ]]
  then
    echo "first round"
  elif [[ $i = 2 ]]
  then
    echo "second round"
  else
    echo "round $i"
  fi
done
```

Write a script that will check if "black" is inside the color array we used before and will print "yes" or "no" at the end.

```
color=(red green blue) # the color array
```

Commands in parallel

Run command in the background, add **&**

```
command1 &
command2 &
wait # if you want to wait until commands are done
```

Background commands in a for loop, with maximum number of threads

```
count=0
for i in {1..10}
do
(
command1
command2
) &
letcount+=1

# () for expression to be interpreted as mathematical operation
# % modulo, returns the remainder of the division

if [[ $((count%4)) -eq 0 ]] # max 4 threads
then
  wait
fi
```

Special characters

#	comment character, anything after on same line is ignored
$	expansion character (variables)
"text"	protects text from being split into multiple words or arguments
'text'	similar as "text", however prevents special characters meaning
****	escape character, prevents next character to be special
> and <	redirect characters (of input / output of a command)
|	pipe character, sends output of one command to the input of the next

Write a script that will print out exactly **How are * doing $today?**