



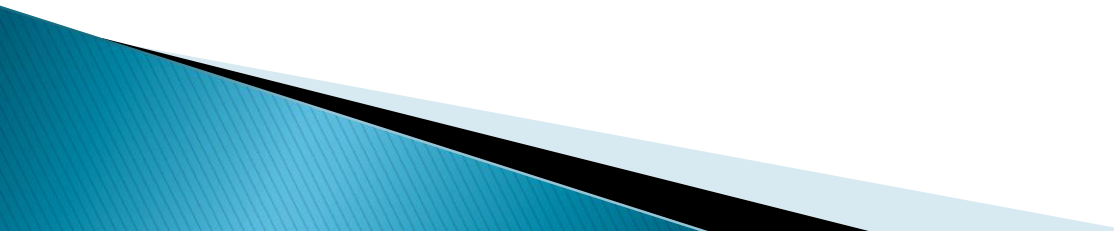
Informatics on High-throughput Sequencing Data

(Summer Course 2020)

Day 7



Agenda

- ▶ **Shell Scripting**
 - ▶ **Variables**
 - ▶ **User Inputs**
 - ▶ **Arithmetic**
 - ▶ **Functions**
 - ▶ **IF statements**
 - ▶ **Loops**
- 

Bash Shell Scripting

- ▶ Bash is a command language interpreter.
- ▶ It is widely available on various operating systems and is a default command interpreter on most GNU/Linux systems.
- ▶ The name is an acronym for the 'Bourne-Again SHell'
- ▶ Scripting allows for an automatic commands execution that would otherwise be executed interactively one-by-one.
- ▶ To see what is your default interpreter execute command:

`echo $SHELL or echo $0`

<https://linuxconfig.org/bash-scripting-tutorial-for-beginners>

Bash Shell Scripting

1. **vi** to create a new file called **task.sh** containing all the commands you want to execute, each on a separate line.
2. Once ready, make your new file executable using **chmod** command with an option **+x**.
3. Lastly, execute your new script by prefixing its name with **./**

Bash Shell Scripting

- ▶ To define your script's interpreter as **Bash**, first locate a full path to its executable binary using **which** command, prefix it with a shebang **#!** and **insert it as the first line of your script.**
 - ▶ `#!/usr/bin/env bash`
 - ▶ `#!/bin/bash`

<https://linuxconfig.org/bash-scripting-tutorial-for-beginners>

Bash Shell Scripting

- ▶ `#!/usr/bin/env bash`
- ▶ `chmod +x myscript.sh`
- ▶ `./myscript.sh`
- ▶ `which bash > myscript.sh + add #!`

<https://linuxconfig.org/bash-scripting-tutorial-for-beginners>

Bash Shell Scripting

- ▶ Another way to execute bash scripts is to call bash interpreter explicitly eg. `$ bash myscript.sh`, hence executing the script **without the need to make the shell script executable and without declaring shebang directly within a shell script.**

Bash Shell Scripting

- ▶ Another way to execute bash scripts is to call bash interpreter explicitly eg. `$ bash myscript.sh`, hence executing the script **without the need to make the shell script executable and without declaring shebang directly within a shell script.**

```
1  #!/bin/bash
2
3  echo "Hello World"
```


Variables

```
1  #!/bin/bash
2
3  greeting="Welcome"
4  user=$(whoami)
5  day=$(date +%A)
6
7  echo "$greeting back $user! Today is $day, which is the best day of the entire week!"
8  echo "Your Bash shell version is: $BASH_VERSION. Enjoy!"
```

- ▶ a=4
- ▶ b=8
- ▶ echo \$ [\$a+\$b]

<https://linuxconfig.org/bash-scripting-tutorial-for-beginners>

User Input

introduction.sh

```
1. #!/bin/bash
2. # Ask the user for their name
3.
4. echo Hello, who am I talking to?
5.
6. read varname
7.
8. echo It\'s nice to meet you $varname
```

<https://ryanstutorials.net/bash-scripting-tutorial/bash-input.php>

Arithmetic

let_example.sh

```
1. #!/bin/bash
2. # Basic arithmetic using let
3.
4. let a=5+4
5. echo $a # 9
6.
7. let "a = 5 + 4"
8. echo $a # 9
9.
10. let a++
11. echo $a # 10
12.
13. let "a = 4 * 5"
14. echo $a # 20
15.
16. let "a = $1 + 30"
17. echo $a # 30 + first command line argument
```

Operator	Operation
----------	-----------

+, -, *, /	addition, subtraction, multiply, divide
-------------	---

var++	Increase the variable var by 1
-------	--------------------------------

var--	Decrease the variable var by 1
-------	--------------------------------

%	Modulus (Return the remainder after division)
---	---

<https://ryanstutorials.net/bash-scripting-tutorial/bash-arithmetic.php>

Arithmetic

expr_example.sh

```
1. #!/bin/bash
2. # Basic arithmetic using expr
3.
4. expr 5 + 4
5.
6. expr "5 + 4"
7.
8. expr 5+4
9.
10. expr 5 \* $1
11.
12. expr 11 % 2
13.
14. a=$( expr 10 - 3 )
15. echo $a # 7
```

<https://ryanstutorials.net/bash-scripting-tutorial/bash-arithmetic.php>

Arithmetic

expansion_example.sh

```
1. #!/bin/bash
2. # Basic arithmetic using double parentheses
3.
4. a=$(( 4 + 5 ))
5. echo $a # 9
6.
7. a=$((3+5))
8. echo $a # 8
9.
10. b=$(( a + 3 ))
11. echo $b # 11
12.
13. b=$(( $a + 4 ))
14. echo $b # 12
15.
16. (( b++ ))
17. echo $b # 13
18.
19. (( b += 3 ))
20. echo $b # 16
21.
22. a=$(( 4 * 5 ))
23. echo $a # 20
```

Arithmetic

length_example.sh

```
1. #!/bin/bash
2. # Show the length of a variable.
3.
4. a='Hello World'
5. echo ${#a} # 11
6.
7. b=4953
8. echo ${#b} # 4
```