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 ./

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

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**.

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**.

```
#!/bin/bash

echo "Hello World"
```

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

Variables

```bash
#!/bin/bash

greeting="Welcome"
user=$(whoami)
day=$(date +%A)

echo "$greeting back $user! Today is $day, which is the best day of the entire week!"
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

```
#!/bin/bash
# Ask the user for their name

echo Hello, who am I talking to?

read varname

echo It\'s nice to meet you $varname
```

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

Arithmetic

let_example.sh

```bash
#!/bin/bash
# Basic arithmetic using let

let a=5+4
echo $a # 9

let "a = 5 + 4"
echo $a # 9

let a++
echo $a # 10

let "a = 4 * 5"
echo $a # 20

let "a = $1 + 30"
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

```
#!/bin/bash
# Basic arithmetic using expr

expr 5 + 4

expr "5 + 4"

expr 5+4

expr 5 \* $1

expr 11 % 2

a=$( expr 10 - 3 )
echo $a # 7
```

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

Arithmetic

expansion_example.sh

```
#!/bin/bash
# Basic arithmetic using double parentheses

a=$(( 4 + 5 ))
echo $a # 9

a=$((3+5))
echo $a # 8

b=$(( a + 3 ))
echo $b # 11

b=$(( $a + 4 ))
echo $b # 12

(( b++ ))
echo $b # 13

(( b += 3 ))
echo $b # 16

a=$(( 4 * 5 ))
echo $a # 20
```

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

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
```

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