# Shell Scripting


BASH

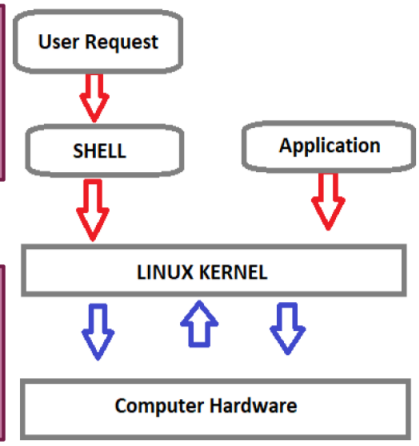
THE BOURNE-AGAIN SHELL

CLARUSWAY©
WAY TO REINVENT YOURSELF

# What is SHELL?

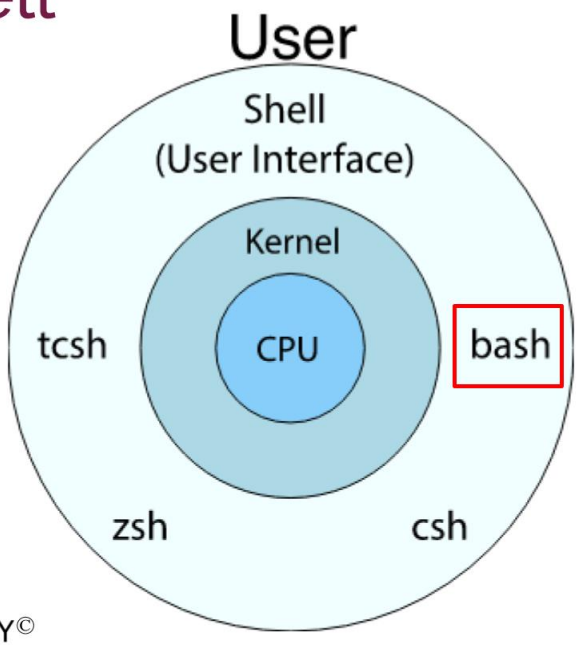
Shell is **a program** that **receives** the user's **commands** and **gives** them to the **operating system to process** and displays the output.

The standard Linux shell is both a **command-line interpreter** and a **programming language.**




CLARUSWAY©
WAY TO REINVENT YOURSELF

# Shell




**SH**

**Bourne-Again SHell**


CLARUSWAY©
WAY TO REINVENT YOURSELF

6

# Shell

1. **Bash : Bourne Again shell**
   The standard GNU shell, intuitive and flexible
2. **ksh : Korn shell**
   A superset of the Bourne shell
3. **csh : C shell**
   The syntax of this shell resembles that of the C programming language
4. **tcsh: TENEX C shell**
   A Superset of the common C shell, enhancing user-friendliness and speed
5. **zsh : Z Shell**
   An extended Bourne shell with a large number of improvements, including some features of Bash, ksh, and tcsh.

Common Shell Types

CLAR
WAY TO REINVENT YOURSELF

7

2


CLARUSWAY©
WAY TO REINVENT YOURSELF

# The Bash prompt

- A prompt is text or symbols used to represent the system's readiness to perform the next command.
- **PS1** variable stands for "Prompt String One" and represents the primary prompt string.

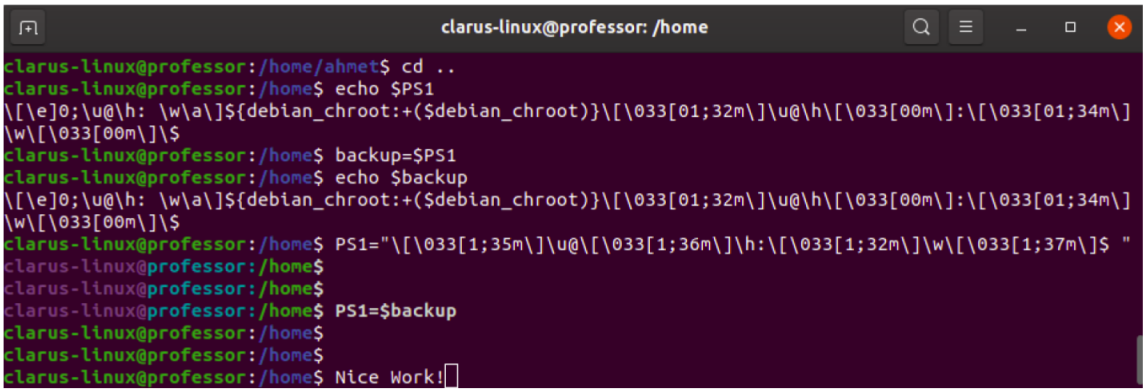
```
user@host:~$
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

9

# The Bash prompt

BASH prompt can be easily changed by modifying bash `PS1` variable.






# The Bash prompt

Bash prompt can be customized by using special characters. Here is the most used characters and their meaning.

| Special Character | Explanation |
|---|---|
| \d | The date in "Weekday Month Date" format (e.g., "Sun Apr 12") |
| \h | the hostname up to the first `.' |
| \H | the hostname |
| \s | the name of the shell, the basename of $0 (the portion following the final slash) |
| \t | the current time in 24-hour HH:MM:SS format |
| \T | the current time in 12-hour HH:MM:SS format |
| \@ | the current time in 12-hour am/pm format |
| \u | the username of the current user |

# The Bash prompt

Bash prompt can be customized by using special characters. Here is the most used characters and their meaning.

| Special Character | Explanation |
|---|---|
| \v | the version of bash (e.g., 2.00) |
| \V | the release of bash, version + patch level (e.g., 2.00.0) |
| \w | the current working directory |
| \W | the basename of the current working directory |
| \! | the history number of this command |



# Homework

How can we make permanent our changes in PS1

# Shell Scripts

**What is Shell Scripting?**

Shell Scripting is an open-source computer program designed to be run by the Unix/Linux shell which could be one of the following:

- The Bourne Shell
- The C Shell
- The Korn Shell
- The GNU Bourne-Again Shell



# Shell Scripts

**What is Shell Scripting?**

- Typical activities that can be done in a shell, such as file manipulation, program execution, and printing text, can also be done with the shell script.

- Lengthy and repetitive sequences of commands can be combined into a single script that can be stored and executed anytime.

# Shell Scripts

```
clarus-linux@professor: ~
clarus-linux@professor:~$ vim class.sh
clarus-linux@professor:~$ chmod +x class.sh
clarus-linux@professor:~$ ./class.sh
Hello World!
clarus-linux@professor:~$
```

```
clarus-linux@professor: ~
#!/bin/bash

echo "Hello World!"

~
~
~
~
"class.sh" 5L, 35C
```

Shebang (#!)


CLARUSWAY©
WAY TO REINVENT YOURSELF

17

# Shell Scripts

```
clarus-linux@professor: ~
clarus-linux@professor:~$ vim class.sh
clarus-linux@professor:~$ chmod +x class.sh
clarus-linux@professor:~$ ./class.sh
Hello World!
clarus-linux@professor:~$
```

```
clarus-linux@professor: ~
#!/bin/bash

echo "Hello World!"

~
~
~
~
"class.sh" 5L, 35C
```

chmod

CLARUSWAY©
WAY TO REINVENT YOURSELF

18

# Shell Scripts

```
clarus-linux@professor: ~
clarus-linux@professor:~$ vim class.sh
clarus-linux@professor:~$ chmod +x class.sh
clarus-linux@professor:~$ ./class.sh
Hello World!
clarus-linux@professor:~$
```

```
clarus-linux@professor: ~
#!/bin/bash

echo "Hello World!"

~
~
~
~
"class.sh" 5L, 35C
```

"./"



# Shell Scripts

```
clarus-linux@professor: ~
#!/bin/bash

echo "Hello World"
date
echo "Waov i learnt one more thing!"
~
~
                                    5,36          All
```

```
clarus-linux@professor: ~
clarus-linux@professor:~$ vi test.sh
clarus-linux@professor:~$
clarus-linux@professor:~$
clarus-linux@professor:~$
clarus-linux@professor:~$ chmod +x test.sh
clarus-linux@professor:~$
```

# Exercise 1

1. Create a script named: "**my-first-script.sh**"

   It should print: "**This is my first script.**"
2. Make the script executable.
3. Execute the script.



Pear Deck Interactive Slide
Do not remove this bar

21

# Homework

Create an environment that you don't need to provide "./" before your scripts while executing them.


CLARUSWAY
WAY TO REINVENT YOURSELF

22

# Variables

- A variable is pointer to the actual data. The shell enables us to create, assign, and delete variables.
- The name of a variable can contain only letters (a to z or A to Z), numbers ( 0 to 9) or the underscore character (_) and beginning with a letter or underscore character.
- The reason you cannot use other characters such as !, *, or - is that these characters have a special meaning for the shell.

```
$VARIABLE=value
$echo $VARIABLE
value
$
$my_var=my_value
$echo $my_var
my_value
$
$my-var=my-value
my-var=my-value: command not
found
$
$myvar?=my-value
myvar?=my-value: command not
found
```



# Variables

**variable=value**

This is one of those areas where formatting is important. Note there is no space on either side of the equals ( = ) sign. We also leave off the $ sign from the beginning of the variable name when setting it.

```
sampledir=/etc
ls $sampledir
```

```
$ myvar='Hello World'
$ echo $myvar
Hello World
$ newvar="More $myvar"
$ echo $newvar
More Hello World
$ newvar='More $myvar'
$ echo $newvar
More $myvar
$
```

# Console input

**read [variable-name]**

```
#!/bin/bash

echo "Enter your name: "
read name
echo Hello $name
~
~
```

```
[ec2-user@ip-172-31-36-108 ~]$ ./run.sh
Enter your name:
Raymond
Hello Raymond
[ec2-user@ip-172-31-36-108 ~]$
```

CLARUSWAY© WAY TO REINVENT YOURSELF



# Console input

**read**

```
#!/bin/bash

read -p "Enter Your Name: "  username
echo "Welcome $username!"
```

```
#!/bin/bash

read -s -p "Enter Password: " pswd
echo $pswd
```

```
#!/bin/bash

read -sp "Enter Password: " pswd
echo $pswd
```

```
#!/bin/bash

echo What cars do you like?

read car1 car2 car3

echo Your first car was: $car1
echo Your second car was: $car2
echo Your third car was: $car3
```

CLARUSWAY© WAY TO REINVENT YOURSELF

# Command Line Arguments

**$0** - The name of the Bash script.
**$1 - $9** - The first 9 arguments to the Bash script.
**$#** - How many arguments were passed to the Bash script.
**$@** - All the arguments supplied to the Bash script.
**$?** - The exit status of the most recently run process.
**$$** - The process ID of the current script.
**$USER** - The username of the user running the script.
**$HOSTNAME** - The hostname of the machine the script is running on.
**$SECONDS** - The number of seconds since the script was started.
**$RANDOM** - Returns a different random number each time is it referred to.
**$LINENO** - Returns the current line number in the Bash script.




# Command Line Arguments

# Simple Arithmetic

**expr** command print the value of expression to standard output.

**expr item1 operator item2**

**let** is a builtin function of Bash that helps us to do simple arithmetic. It is similar to **expr** except instead of printing the answer it saves the result to a variable.

**let <arithmetic expression>**

We can also evaluate arithmetic expression with double parentheses.

**$((arithmetic expression))**



# Arithmetic Expressions

**expr item1 operator item2**

```
#!/bin/bash
first_number=8
second_number=2

echo "SUM="`expr $first_number + $second_number`
echo "SUB="`expr $first_number - $second_number`
echo "MUL="`expr $first_number \* $second_number`
echo "DIV="`expr $first_number / $second_number`
```

```
$ chmod +x cal.sh
$ ./cal.sh
SUM=10
SUB=6
MUL=16
DIV=4
```

# Arithmetic Expressions

`let [expression]`

```
#!/bin/bash

number1=8
number2=2

let total=number1+number2
let diff=number1-number2
let mult=number1*number2
let div=number1/number2

echo "Total = $total"
echo "Difference = $diff"
echo "Multiplication = $mult"
echo "Division = $div"
```

```
$ ./run.sh
Total = 10
Difference = 6
Multiplication = 16
Division = 4
```

CLARUSWAY© WAY TO REINVENT YOURSELF



# “num++” “++num” “num--” “--num”

```
#!/bin/bash

number=10
let new_number=number++
echo "Number = $number"
echo "New number = $new_number"

number=10
let new_number=--number
echo "Number = $number"
echo "New number = $new_number"

~
```

```
[ec2-user@ip-172-31-91-206 ~]$ ./run.sh
Number = 11
New number = 10
Number = 9
New number = 9
[ec2-user@ip-172-31-91-206 ~]$
```

CLARUSWAY© WAY TO REINVENT YOURSELF

# Arithmetic Expressions

`$((Expression))`

`((Expression))`

```
#!/bin/bash

number1=8
number2=2

echo "Total = $((number1+number2))"

((total=number1+number2))
echo "Total = $total"
~
```

```
[ec2-user@ip-172-31-91-206 ~]$ ./run.sh
Total = 10
Total = 10
[ec2-user@ip-172-31-91-206 ~]$
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

33

# Exercise 1

1. Ask user to enter two numbers to variables **num1** and **num2.**
2. Calculate the total of 2 numbers.
3. Print the **total** number and increase it by 1.
4. Print the new value of the **total** number.
5. Subtract **num1** from the **total** number and print result.

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

34

# Exercise 2

1. Create a script named **calculate.sh:**

   Create a variable named **base_value** with default value of **5**

   Request another number from user and assign it to **user_input** variable

   Add **user_value** to the **base_value** and assign it to **total** variable

   Print **total** to the screen with the message "**Total value is:** "
2. Make the script executable.
3. Execute the script.


Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar 35