Linux Plus for AWS and DevOps

Session - 6

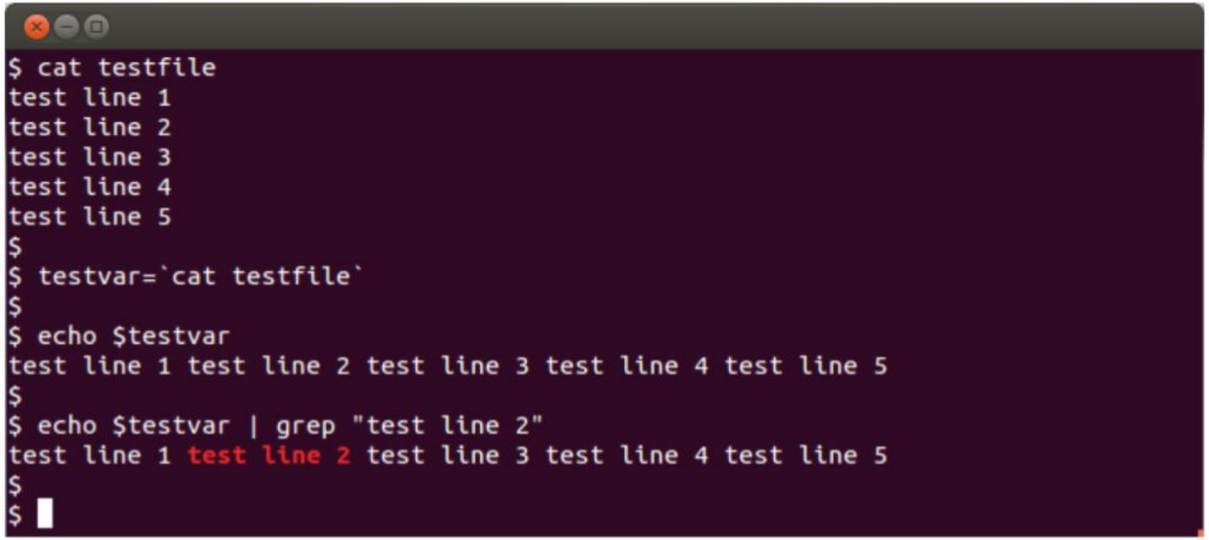
CLARUSWAY
WAY TO REINVENT YOURSELF

Shell Scripting

$_ BASH
THE BOURNE-AGAIN SHELL

Table of Contents

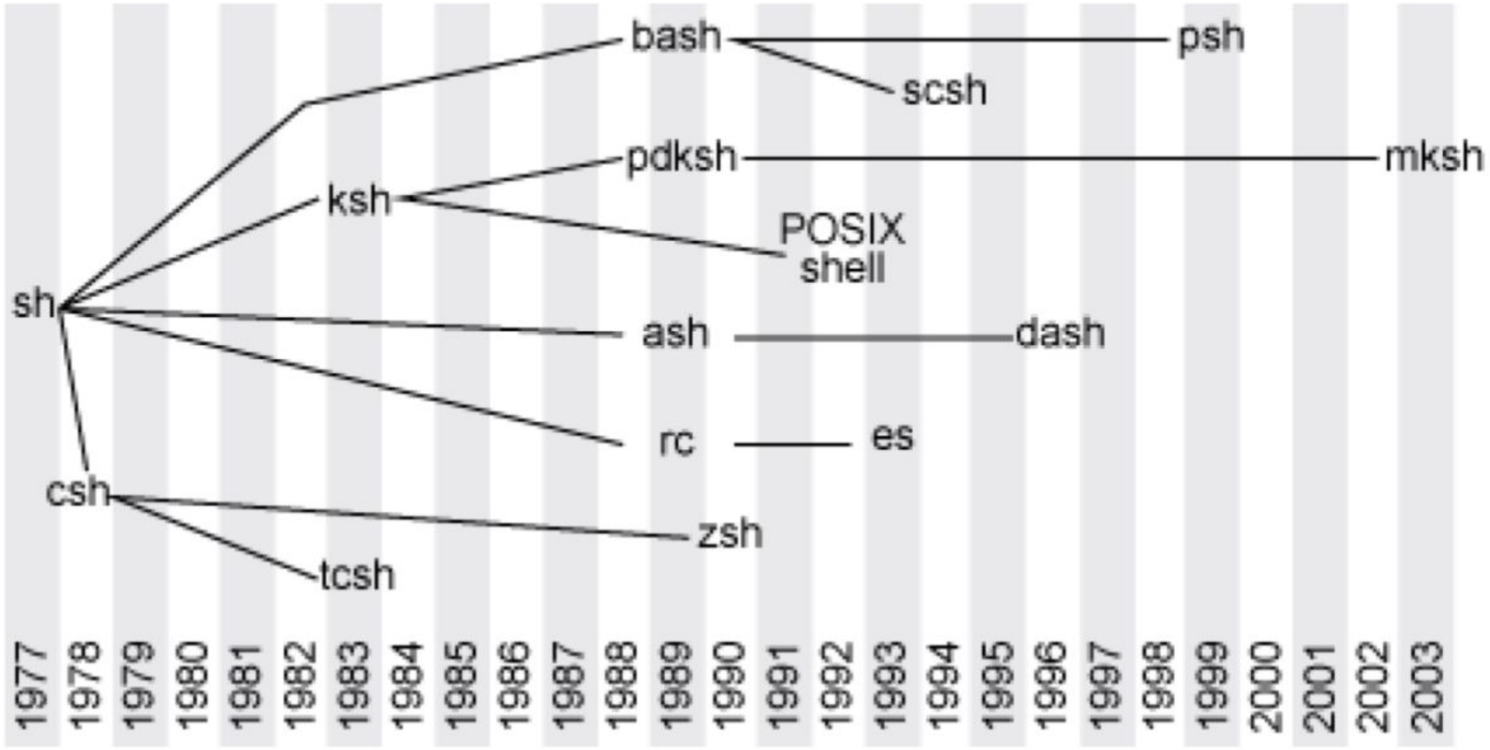
Shell



```
$ cat testfile
test line 1
test line 2
test line 3
test line 4
test line 5
$
$ testvar=`cat testfile`
$
$ echo $testvar
test line 1 test line 2 test line 3 test line 4 test line 5
$
$ echo $testvar | grep "test line 2"
test line 1 test line 2 test line 3 test line 4 test line 5
$
$ 
```

A Unix shell is both a command interpreter and a programming language.

Shell


bash
psh
scsh
pdksh
mksh
ksh
POSIX shell
sh
ash
dash
rc
es
csh
zsh
tcsh
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003

Shell

1. **Bash : Bourne Again shell**
   The standard GNU shell, intuitive and flexible

2. **ksh : Korn shell**
   A superset of the Bourne shell

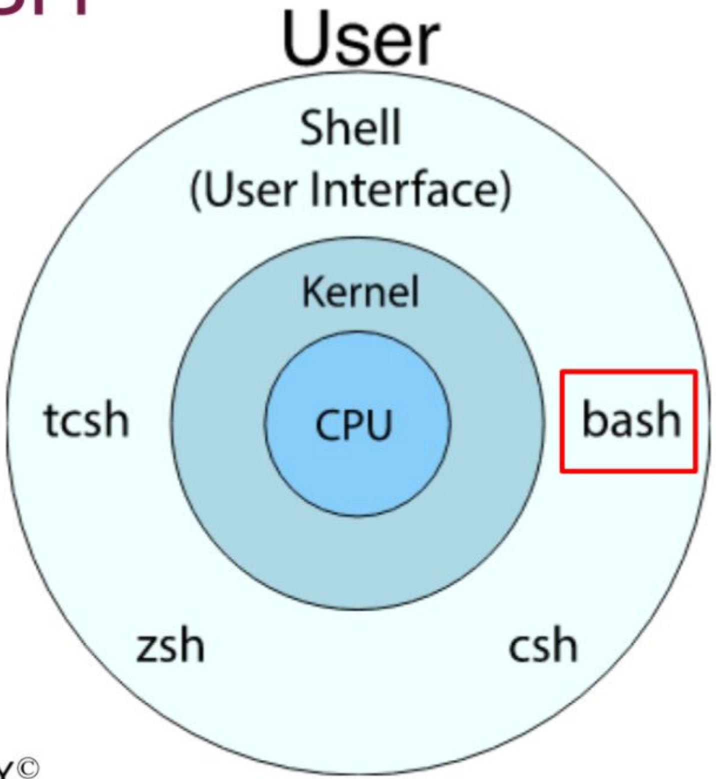
3. **csh : C shell**
   The syntax of this shell resembles that of the C programming language

4. **tcsh: TENEX C shell**
   A Superset of the common C shell, enhancing user-friendliness and speed

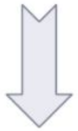
5. **zsh : Z Shell**
   An extended Bourne shell with a large number of improvements, including some features of Bash, ksh, and tcsh.

Common Shell Types

BASH



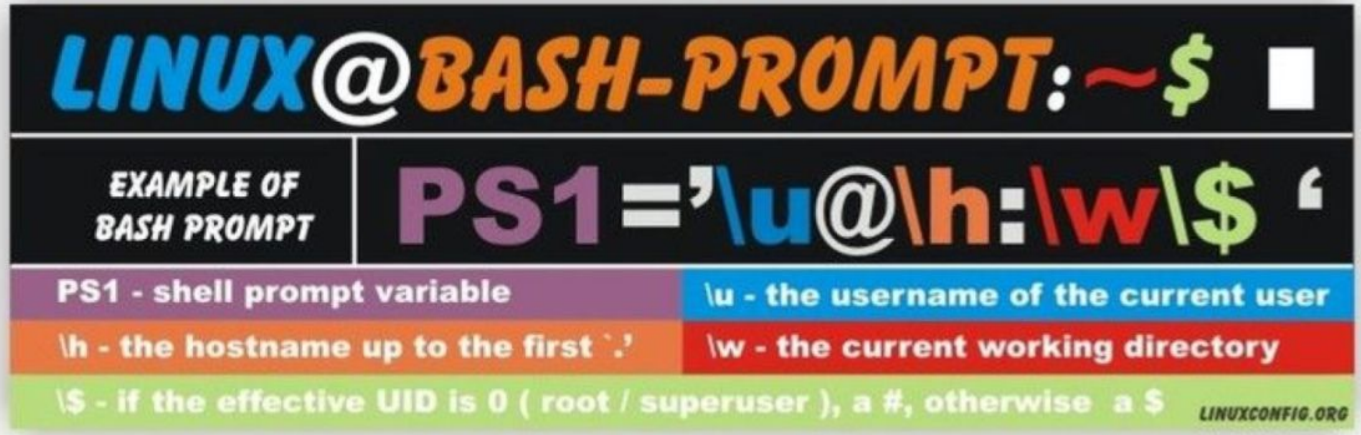
SH



Bourne-Again SHell

CLARUSWAY©
WAY TO REINVENT YOURSELF

2



LINUX@BASH-PROMPT:~$

EXAMPLE OF BASH PROMPT

```
PS1='\u@\h:\w\$ '
```

PS1 - shell prompt variable

\u - the username of the current user

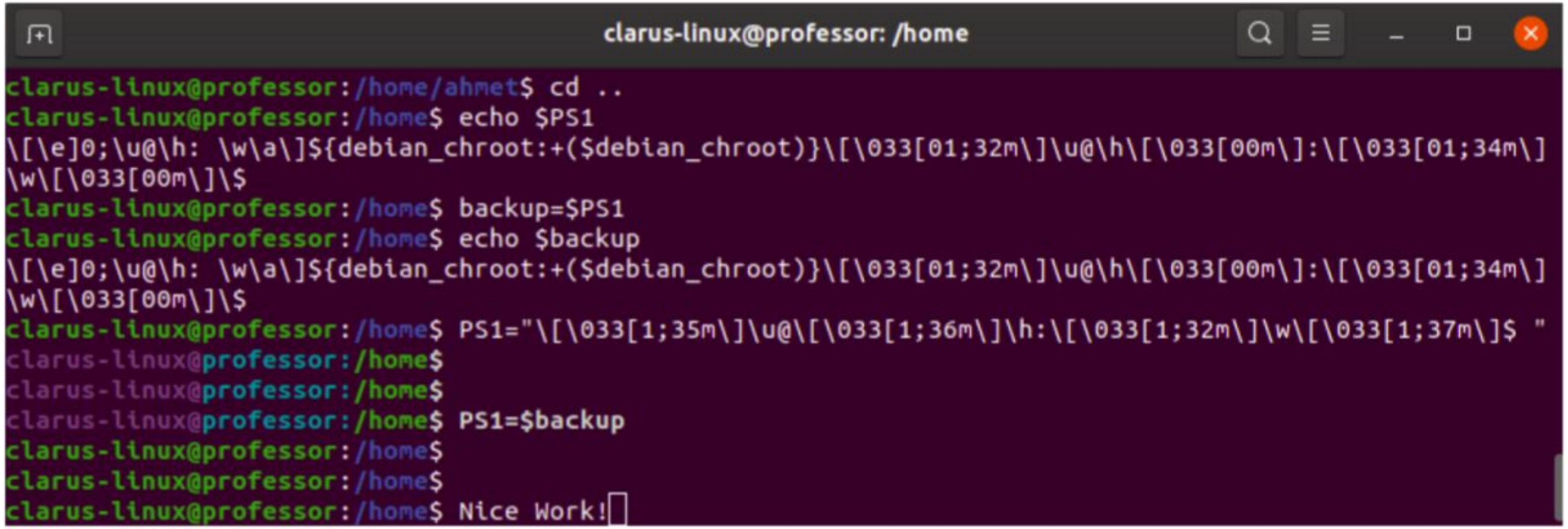
\h - the hostname up to the first `.'

\w - the current working directory

\$ - if the effective UID is 0 (root / superuser), a #, otherwise a $

LINUXCONFIG.ORG

The Bash prompt

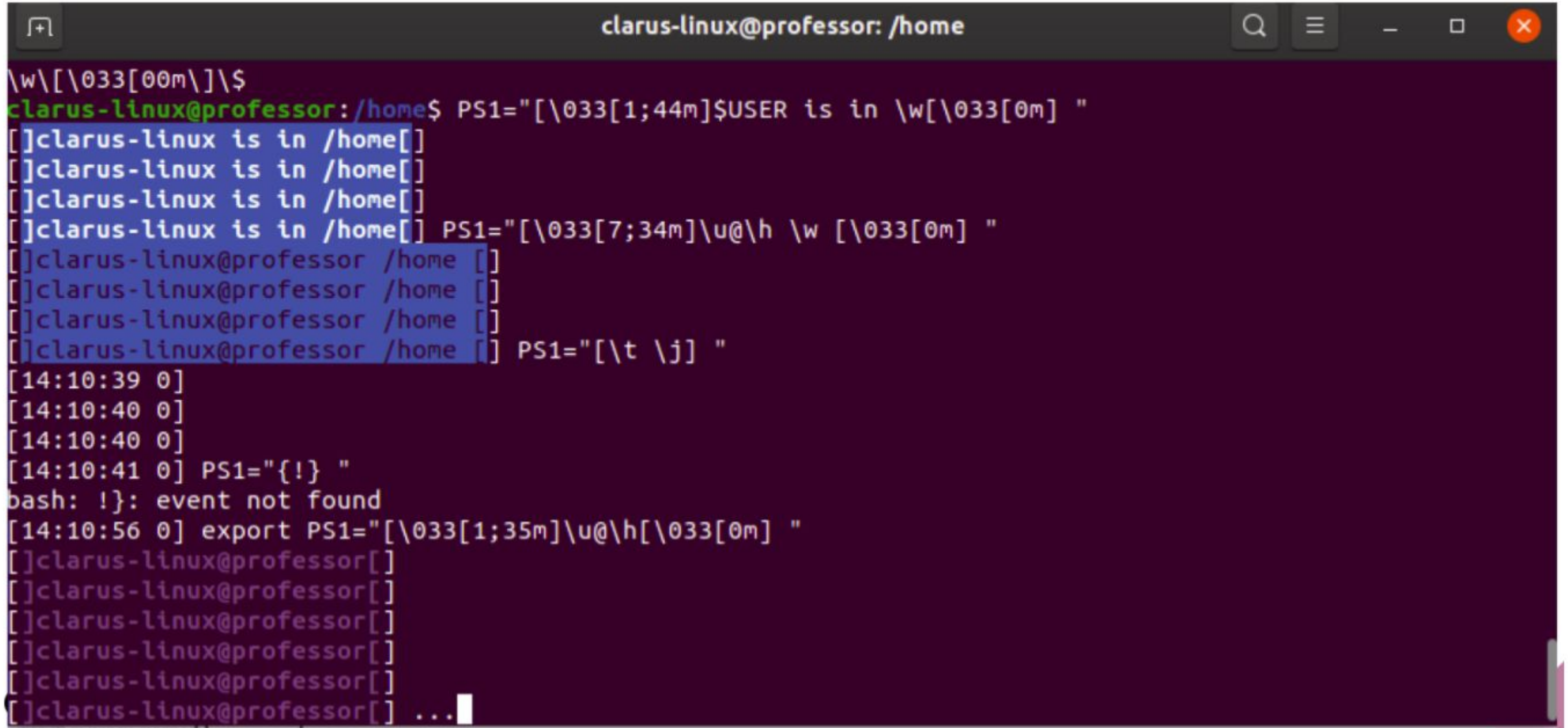


```
clarus-linux@professor:/home/ahmet$ cd ..
clarus-linux@professor:/home$ echo $PS1
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]
\w\[\033[00m\]\$
clarus-linux@professor:/home$ backup=$PS1
clarus-linux@professor:/home$ echo $backup
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]
\w\[\033[00m\]\$
clarus-linux@professor:/home$ PS1="\[\033[1;35m\]\u@\[\033[1;36m\]\h:\[\033[1;32m\]\w\[\033[1;37m\]$ "
clarus-linux@professor:/home$
clarus-linux@professor:/home$
clarus-linux@professor:/home$ PS1=$backup
clarus-linux@professor:/home$
clarus-linux@professor:/home$
clarus-linux@professor:/home$ Nice Work!
```

The Bash prompt

\d	the date in "Weekday Month Date" format (e.g., "Sun Apr 12")
\h	the hostname up to the first `.'
\H	the hostname
\j	the number of jobs currently managed by the shell
\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
\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
\#	the command number of this command
\$	if the effective UID is 0, a #, otherwise a $

The Bash prompt


clarus-linux@professor: /home

```
\w\[\033[00m\]\$
clarus-linux@professor:/home$ PS1="[\033[1;44m]$USER is in \w[\033[0m] "
[]clarus-linux is in /home[]
[]clarus-linux is in /home[]
[]clarus-linux is in /home[]
[]clarus-linux is in /home[] PS1="[\033[7;34m]\u@\h \w [\033[0m] "
[]clarus-linux@professor /home []
[]clarus-linux@professor /home []
[]clarus-linux@professor /home []
[]clarus-linux@professor /home [] PS1="[\t \j] "
[14:10:39 0]
[14:10:40 0]
[14:10:40 0]
[14:10:41 0] PS1="{!} "
bash: !}: event not found
[14:10:56 0] export PS1="[\033[1;35m]\u@\h[\033[0m] "
[]clarus-linux@professor[]
[]clarus-linux@professor[]
[]clarus-linux@professor[]
[]clarus-linux@professor[]
[]clarus-linux@professor[]
[]clarus-linux@professor[] ...
```

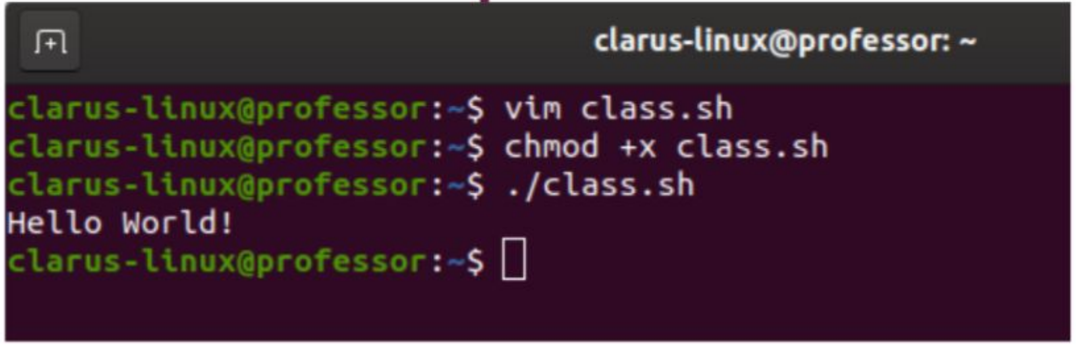
Homework

How can we make permanent our changes in PS1

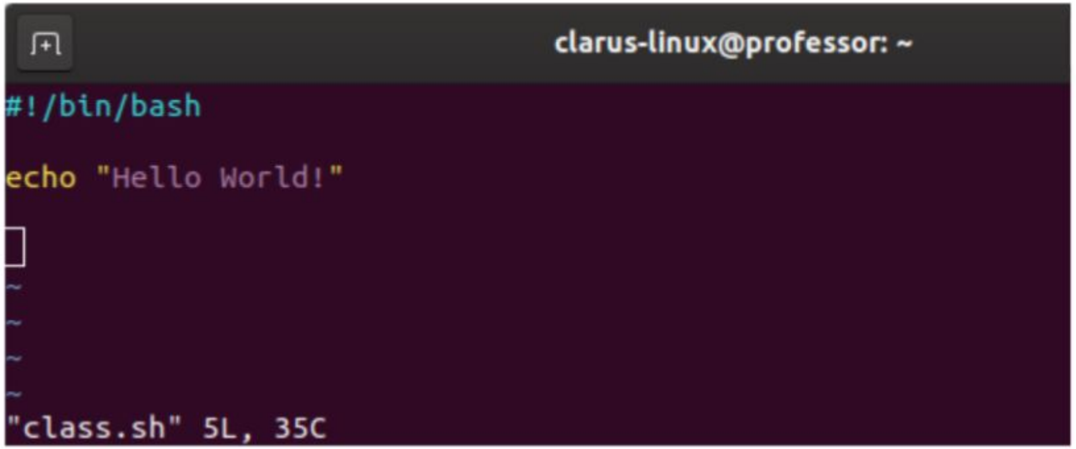
3

Shell Scripts

Shell Scripts



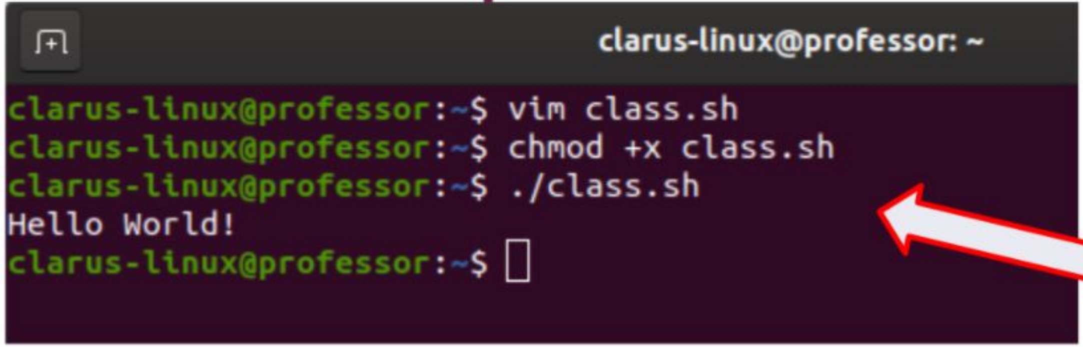
```
clarus-linux@professor:~$ vim class.sh
clarus-linux@professor:~$ chmod +x class.sh
clarus-linux@professor:~$ ./class.sh
Hello World!
clarus-linux@professor:~$ 
```



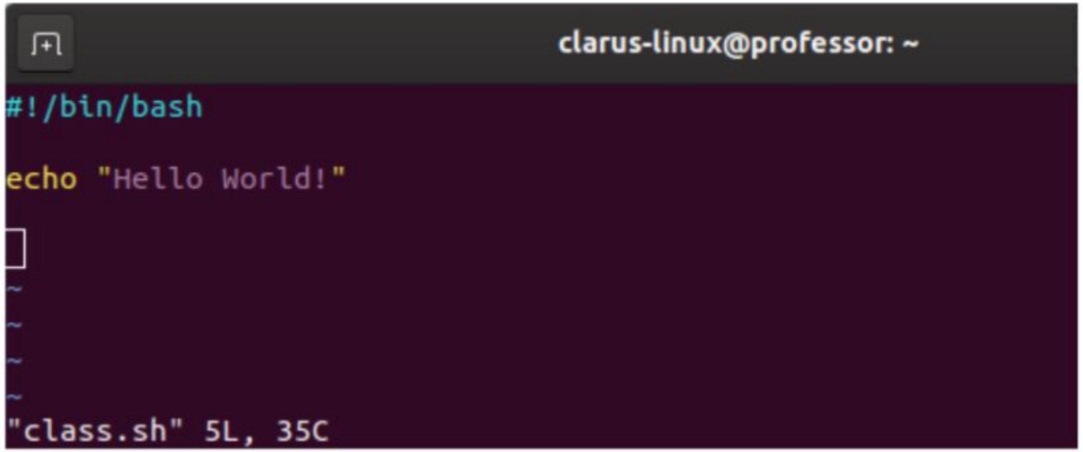
```
#!/bin/bash

echo "Hello World!"

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

Shebang (#!)

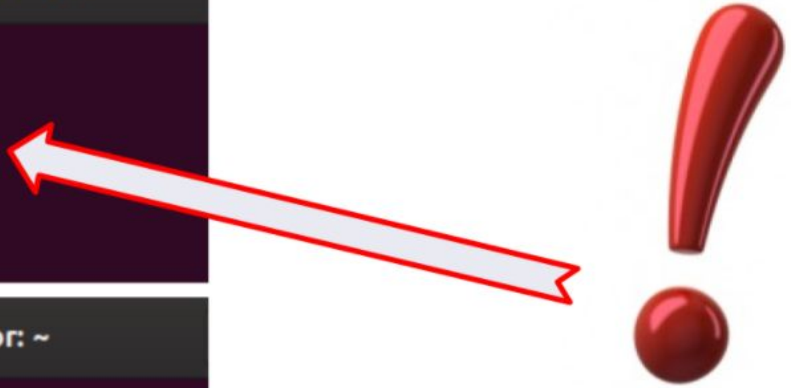
Shell Scripts



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

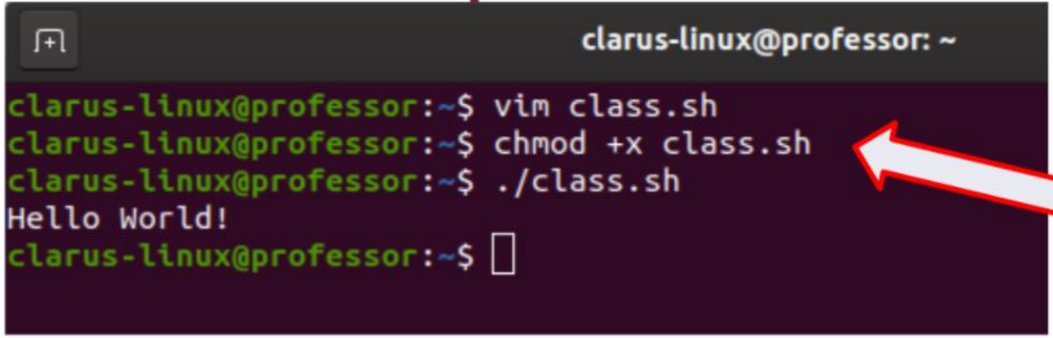


```
#!/bin/bash

echo "Hello World!"

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

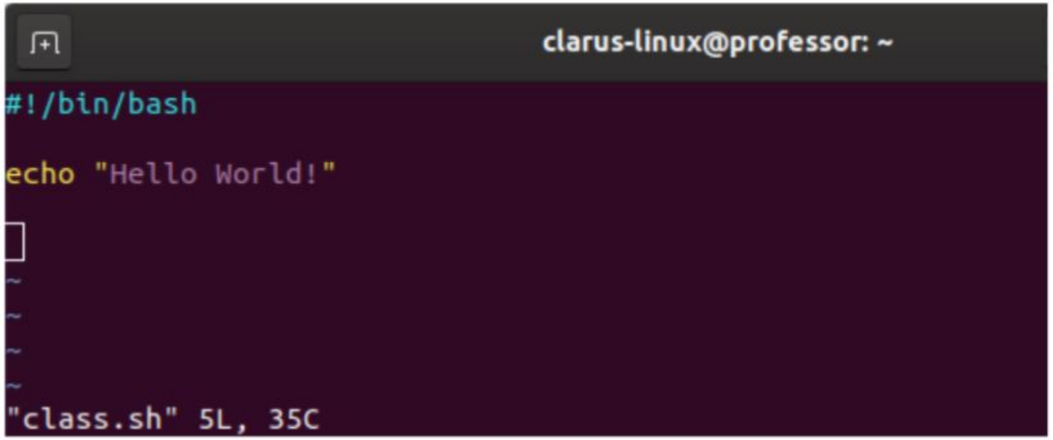


"./"

Shell Scripts



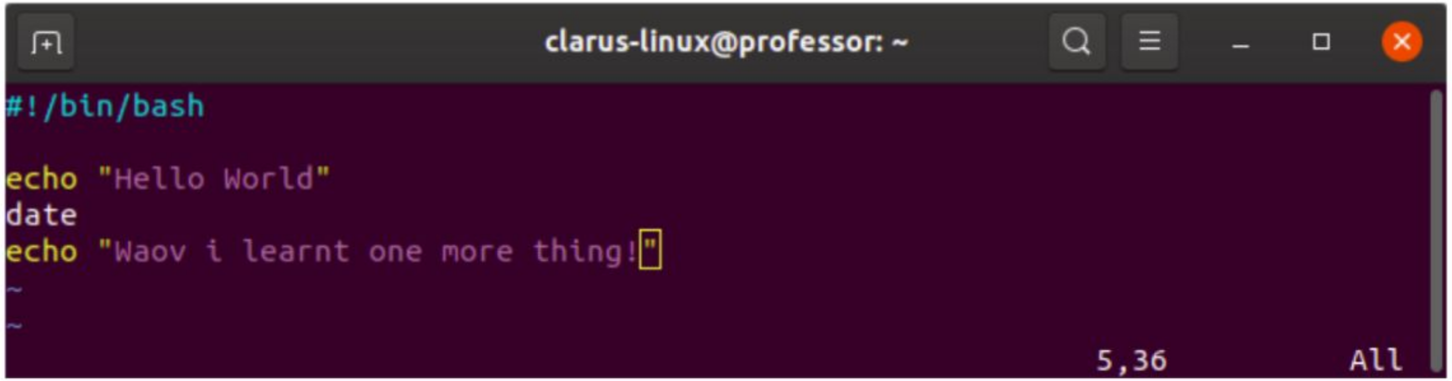
```
clarus-linux@professor: ~
clarus-linux@professor:~$ vim class.sh
clarus-linux@professor:~$ chmod +x class.sh
clarus-linux@professor:~$ ./class.sh
Hello World!
clarus-linux@professor:~$
```



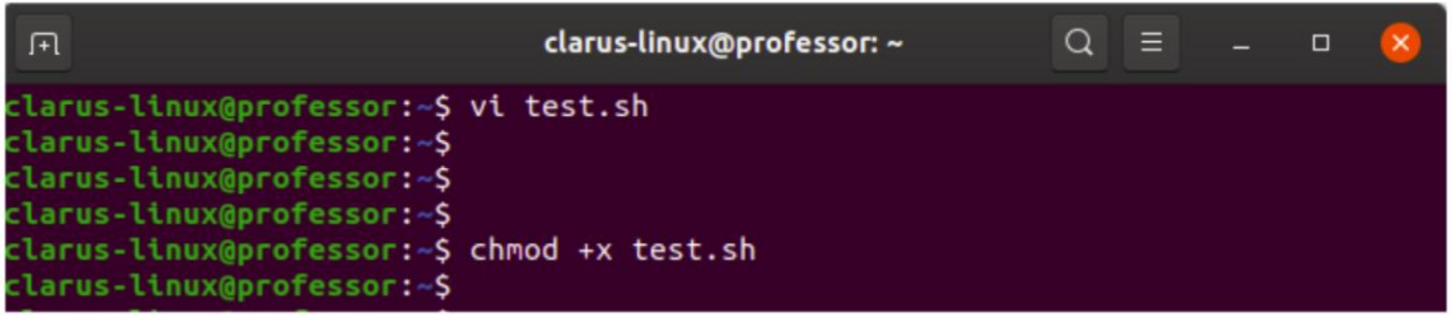
```
clarus-linux@professor: ~
#!/bin/bash

echo "Hello World!"


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

chmod

Shell Scripts


clarus-linux@professor: ~

```
#!/bin/bash

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

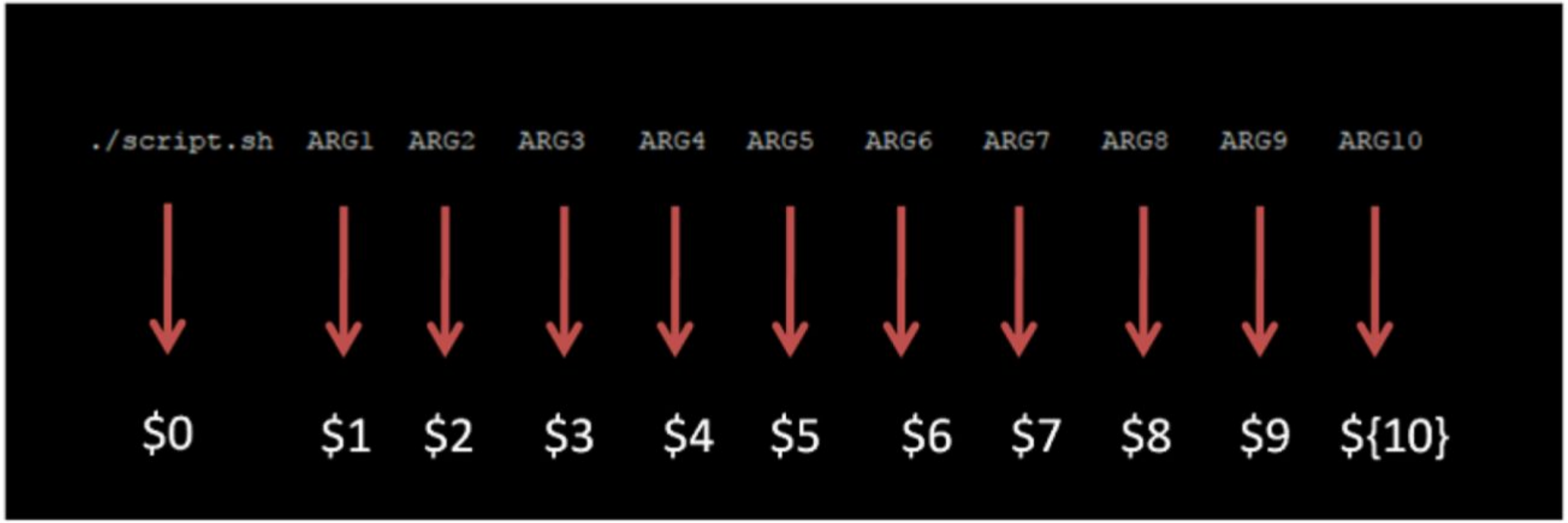

clarus-linux@professor: ~

```
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:~$
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

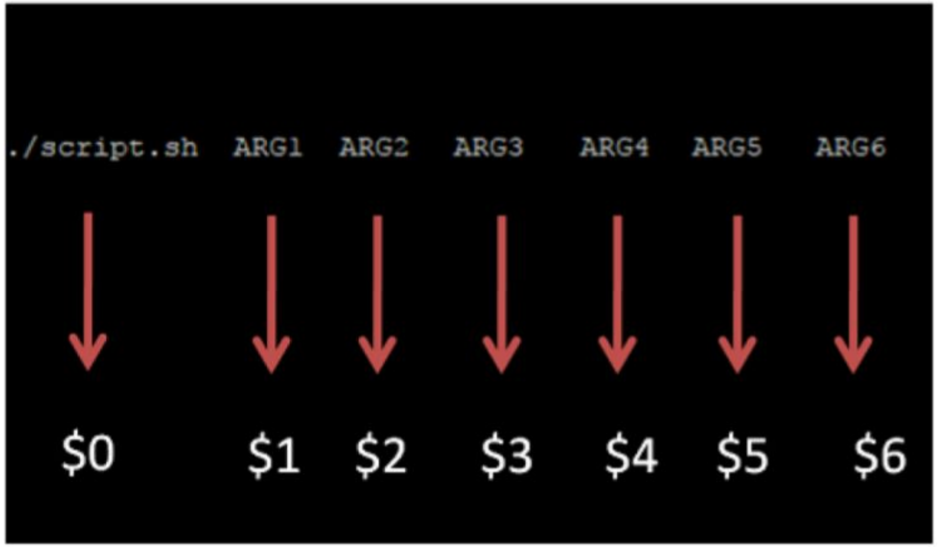
Shell Scripts

Command Line Arguments


./script.sh ARG1 ARG2 ARG3 ARG4 ARG5 ARG6 ARG7 ARG8 ARG9 ARG10
$0 $1 $2 $3 $4 $5 $6 $7 $8 $9 ${10}

Shell Scripts

Command Line Arguments



```
./script.sh ARG1 ARG2 ARG3 ARG4 ARG5 ARG6
```

```
#!/bin/bash

cp $1 $2

echo Details for $2
ls -lh $2
```

Shell Scripts

$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.

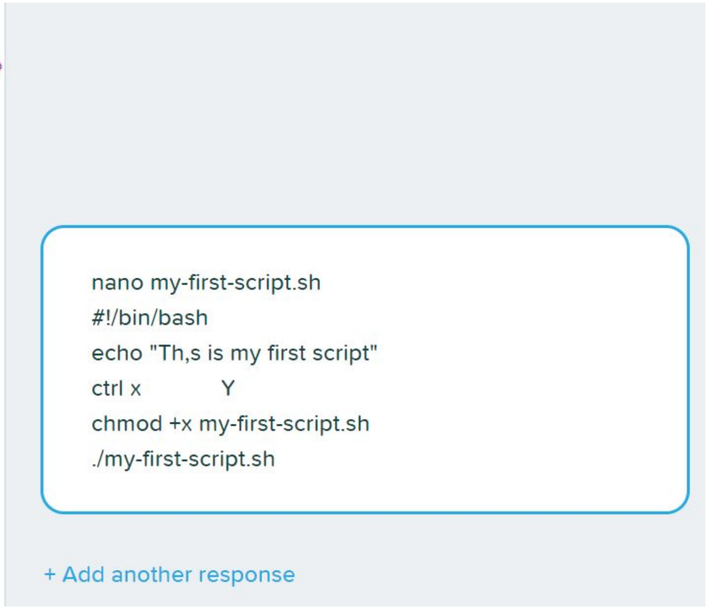
"env"

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.



```
nano my-first-script.sh
#!/bin/bash
echo "Th,s is my first script"
ctrl x            Y
chmod +x my-first-script.sh
./my-first-script.sh
```

Homework

Create an environment that you don’t need to provide “./” before your scripts while executing them.

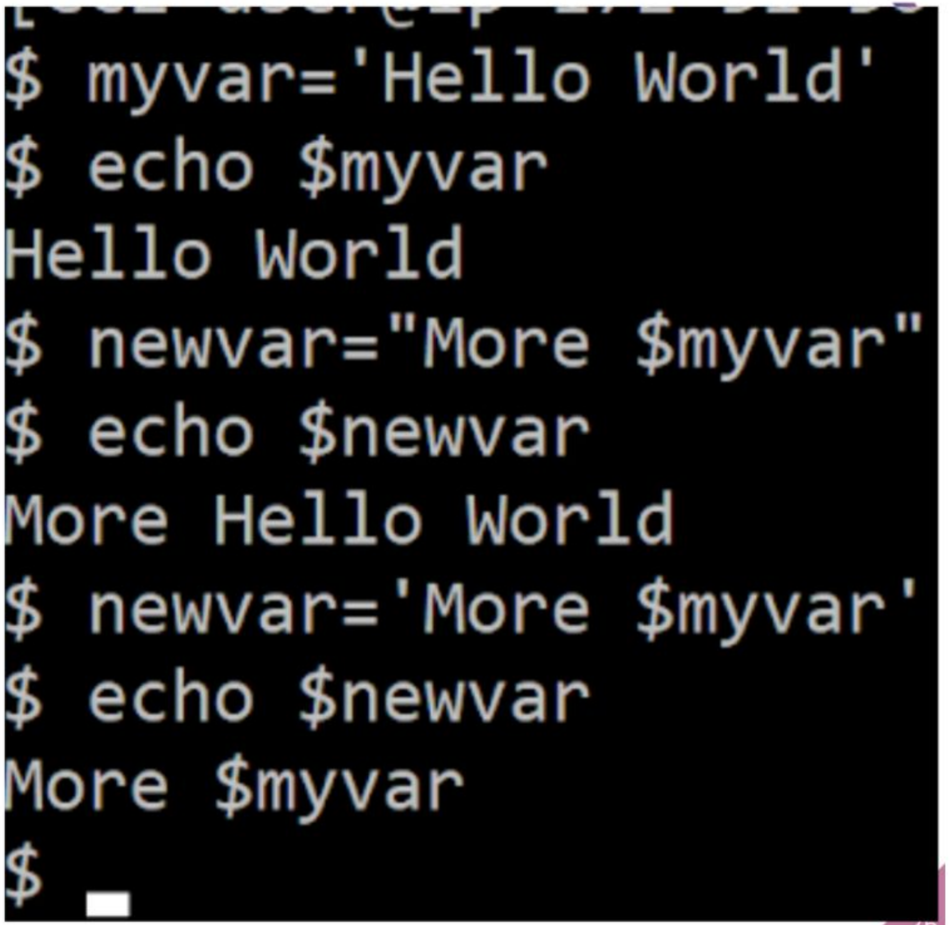
Variables

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
```

CLARUSWAY©
WAY TO REINVENT YOURSELF



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

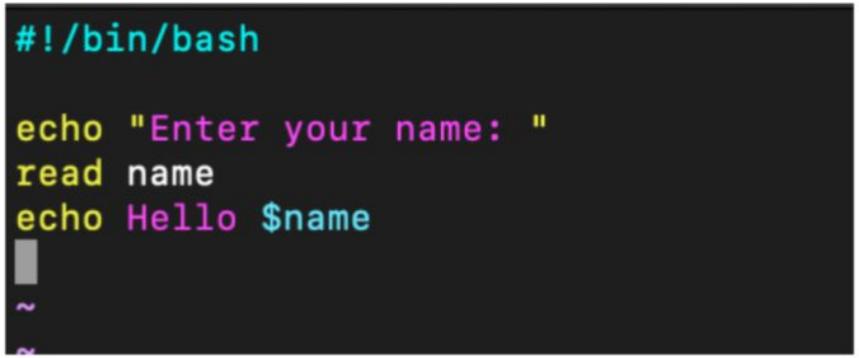
Homework

<u>Research question:</u>

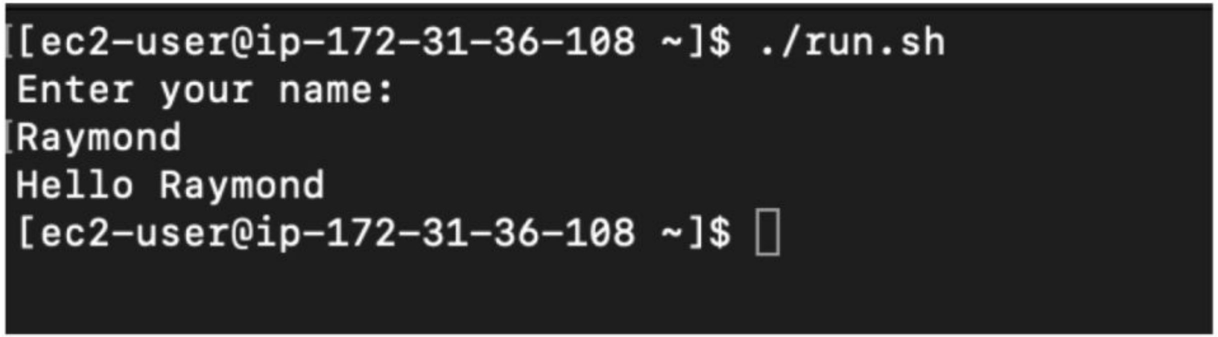
Is it possible to use arrays in Bash scripting? If so, how?

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
```

Simple Arithmetic

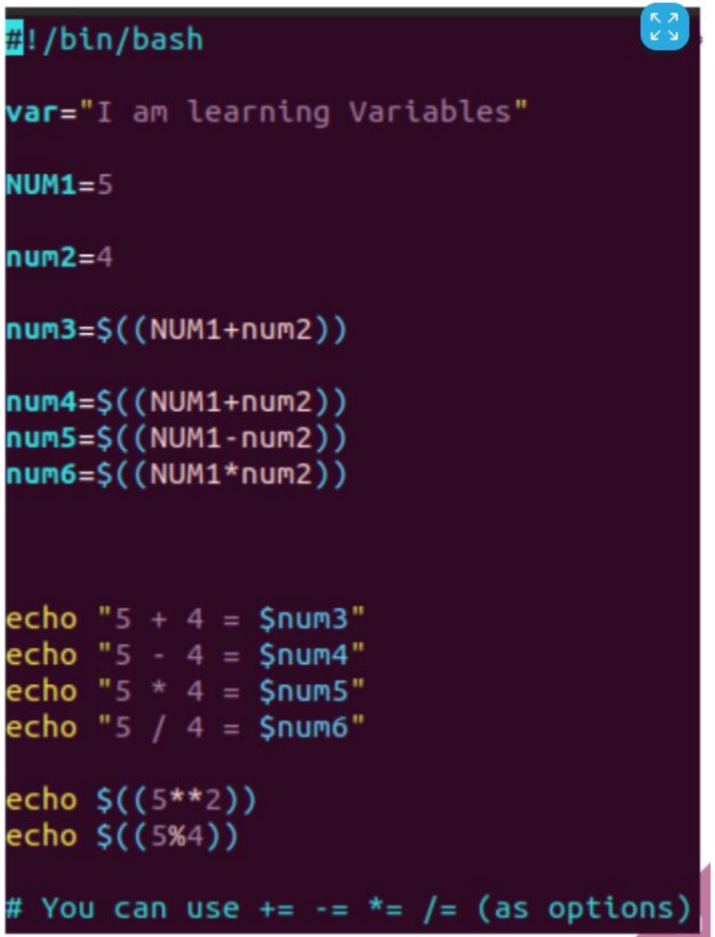
let expression
Make a variable equal to an expression.

expr expression
print out the result of the expression.

$((expression))
Return the result of the expression.

${#var}
Return the length of the variable var.

CLARUSWAY©
WAY TO REINVENT YOURSELF



```
#!/bin/bash

var="I am learning Variables"

NUM1=5

num2=4

num3=$((NUM1+num2))

num4=$((NUM1+num2))
num5=$((NUM1-num2))
num6=$((NUM1*num2))



echo "5 + 4 = $num3"
echo "5 - 4 = $num4"
echo "5 * 4 = $num5"
echo "5 / 4 = $num6"

echo $((5**2))
echo $((5%4))

# You can use += -= *= /= (as options)
```

Simple Arithmetic

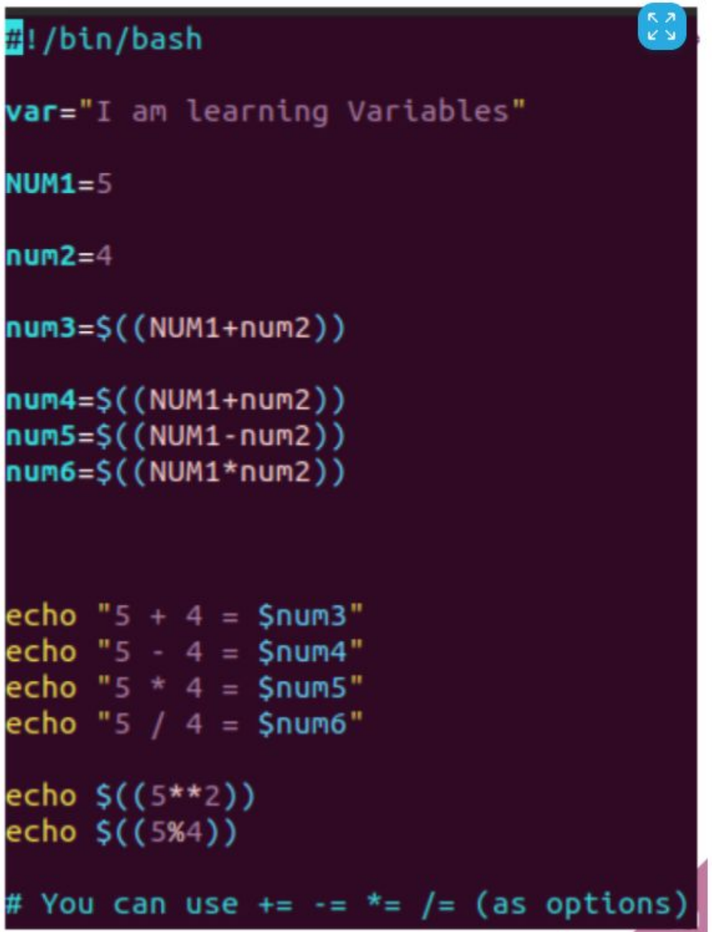
let expression
Make a variable equal to an expression.

expr expression
print out the result of the expression.

$((expression))
Return the result of the expression.

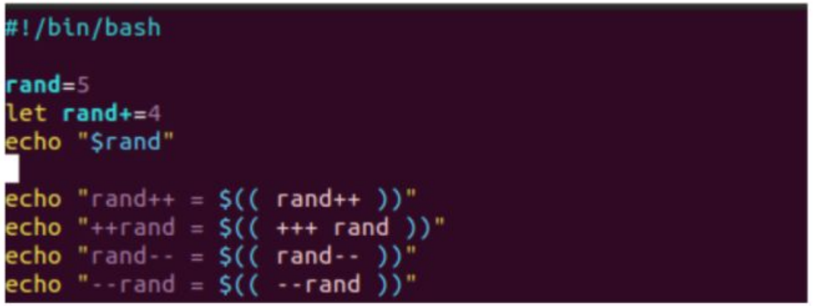
${#var}
Return the length of the variable var.

CLARUSWAY©
WAY TO REINVENT YOURSELF

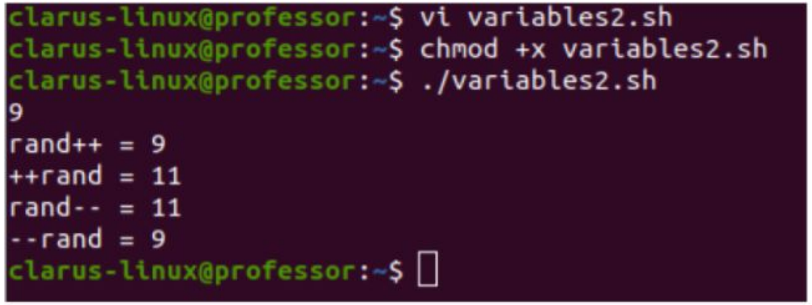


```
#!/bin/bash

var="I am learning Variables"

NUM1=5

num2=4

num3=$((NUM1+num2))

num4=$((NUM1+num2))
num5=$((NUM1-num2))
num6=$((NUM1*num2))



echo "5 + 4 = $num3"
echo "5 - 4 = $num4"
echo "5 * 4 = $num5"
echo "5 / 4 = $num6"

echo $((5**2))
echo $((5%4))

# You can use += -= *= /= (as options)
```

Simple Arithmetic



```
#!/bin/bash

rand=5
let rand+=4
echo "$rand"

echo "rand++ = $(( rand++ ))"
echo "++rand = $(( +++ rand ))"
echo "rand-- = $(( rand-- ))"
echo "--rand = $(( --rand ))"
```

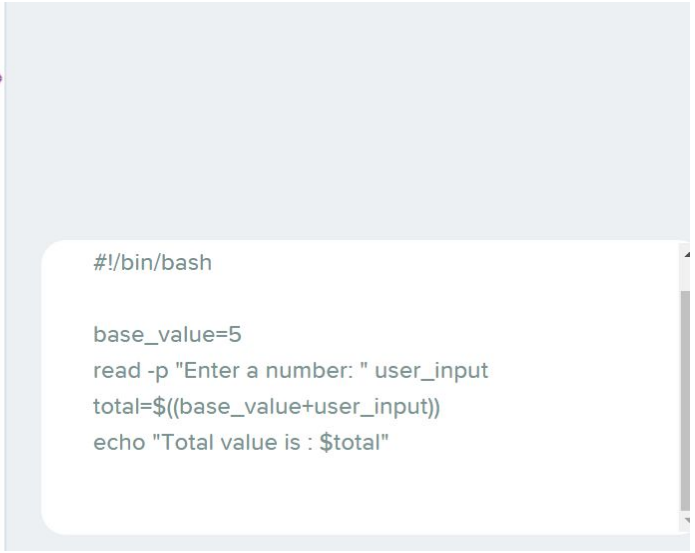


```
clarus-linux@professor:~$ vi variables2.sh
clarus-linux@professor:~$ chmod +x variables2.sh
clarus-linux@professor:~$ ./variables2.sh
9
rand++ = 9
++rand = 11
rand-- = 11
--rand = 9
clarus-linux@professor:~$ 
```

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)

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.



```
#!/bin/bash

base_value=5
read -p "Enter a number: " user_input
total=$((base_value+user_input))
echo "Total value is : $total"
```

Homework

Write a script for login to AWS ec2 instance.

- Get the ip address from user.
- Inform the user before login.
- Design a simple gui by using regular characters!

Run the script on GIT Bash.