Linux Plus for AWS and DevOps

CLARUSWAY
WAY TO REINVENT YOURSELF

Linux Environment Variables

CLARUSWAY
WAY TO REINVENT YOURSELF

Table of Contents

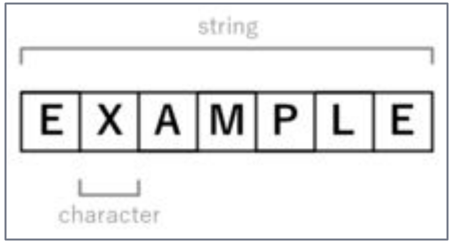
1

What are Environment variables?

CLARUSWAY
WAY TO REINVENT YOURSELF

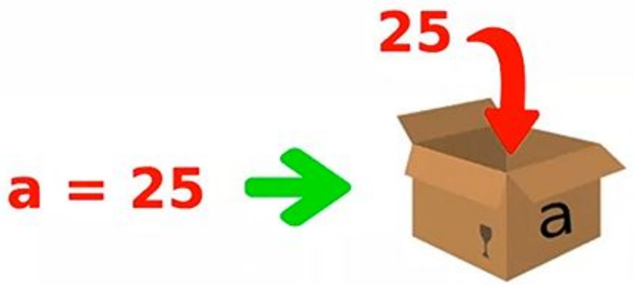
What are Variables?

- A variable is nothing more than a pointer to the actual data.
- A variable is a character string to which we assign a value.
- The value assigned could be a number, text, filename, device, or any other type of data.
- Strings are typically made up of characters, and are often used to store human-readable data, such as words or sentences.



CLARUSWAY
WAY TO REINVENT YOURSELF

Variables



Variables are used to store for data values

What are Environment variables?

Environment variables are variables available to your program/application dynamically during runtime. The value of these variables can come from a range of sources — text files, third-party secret managers, calling scripts, etc.

An environment variable is a **dynamic-named** value that can **affect the way running processes will behave** on a computer. It can be created, edited, saved, and deleted and give information about the system behavior.

What's important here is the fact that the value of these environment variables is not hardcoded in your program. These are truly dynamic and can be changed based on the environment that your program is running in.

What are Environment variables?

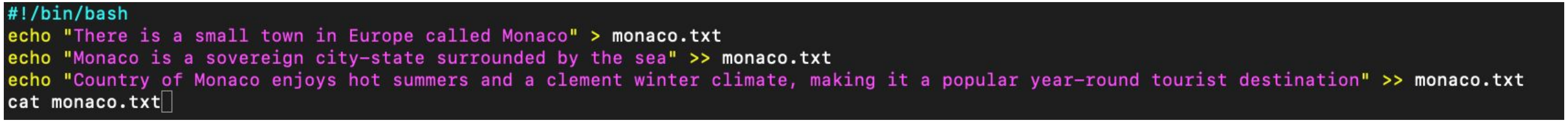
A list of **all specified environment variables** can be viewed entering the **env** command.

There is **nothing special** about variable names, but, by convention, environment variables should have **UPPER CASE** names.

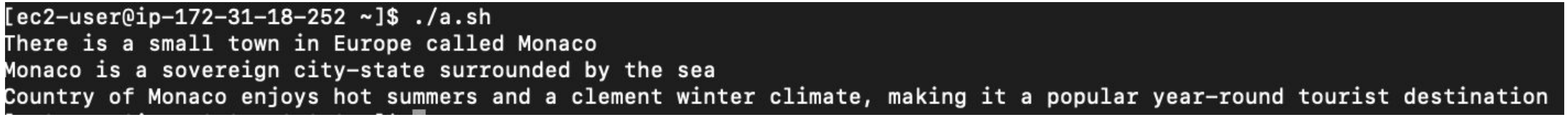
Environment variables allow you to **customize how the system works** and the behavior of the applications on the system.

Example - Environment Variable

Without Environment Variable



```
#!/bin/bash
echo "There is a small town in Europe called Monaco" > monaco.txt
echo "Monaco is a sovereign city-state surrounded by the sea" >> monaco.txt
echo "Country of Monaco enjoys hot summers and a clement winter climate, making it a popular year-round tourist destination" >> monaco.txt
cat monaco.txt
```

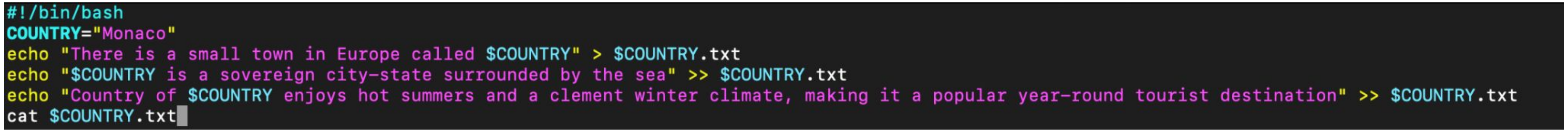


```
[ec2-user@ip-172-31-18-252 ~]$ ./a.sh
There is a small town in Europe called Monaco
Monaco is a sovereign city-state surrounded by the sea
Country of Monaco enjoys hot summers and a clement winter climate, making it a popular year-round tourist destination
```

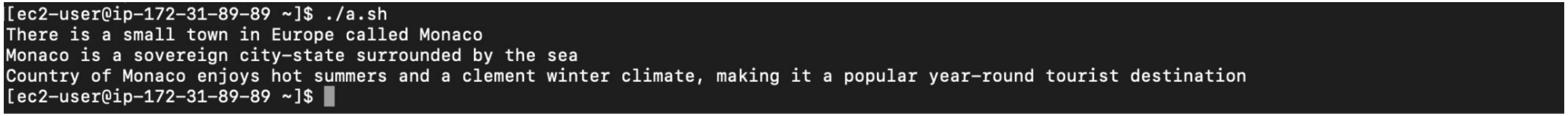
CLARUSWAY
WAY TO REINVENT YOURSELF

Example - Environment Variable

With Environment Variable



```
#!/bin/bash
COUNTRY="Monaco"
echo "There is a small town in Europe called $COUNTRY" > $COUNTRY.txt
echo "$COUNTRY is a sovereign city-state surrounded by the sea" >> $COUNTRY.txt
echo "Country of $COUNTRY enjoys hot summers and a clement winter climate, making it a popular year-round tourist destination" >> $COUNTRY.txt
cat $COUNTRY.txt
```



```
[ec2-user@ip-172-31-89-89 ~]$ ./a.sh
There is a small town in Europe called Monaco
Monaco is a sovereign city-state surrounded by the sea
Country of Monaco enjoys hot summers and a clement winter climate, making it a popular year-round tourist destination
[ec2-user@ip-172-31-89-89 ~]$
```

2

Common Environment Variables

Environment Variable Types

There are two types of variables:

- **Environment variables** are system wide and are inherited by all system processes and shells.
  eg:- use **ENV** command to list environment variables

- **Shell variables** only apply internally to the current shell instance. Once the shell is closed(when terminal is closed), these variables will be lost.
  eg:- use “**set -o posix ; set**” command to list only shell variables

- use **SET** command to list all type of variables within the system

Common Environment Variables

Variable	Description
PATH	This variable contains a **colon (:)-separated list of directorie**s in which your system looks for executable files.
USER	The username
HOME	**Default path** to the **user's home** directory
EDITOR	Path to the program which edits the content of files
UID	User's unique ID
TERM	Default terminal emulator
SHELL	Shell being used by the user
LANG	The current locales settings.

Common Commands

Command	Description
env	The **env** command is a shell command used to display and manipulate environment variables. It is **used to list down** environment variables.
printenv	The command prints all or the specified environment variables.
set	The command sets or unsets shell variables. When used without an argument it will print a list of all variables including environment and shell variables, and shell functions.
unset	The command deletes shell and environment variables.
export	The command sets environment variables.

CLARUSWAY
WAY TO REINVENT YOURSELF

3

Accessing Variable

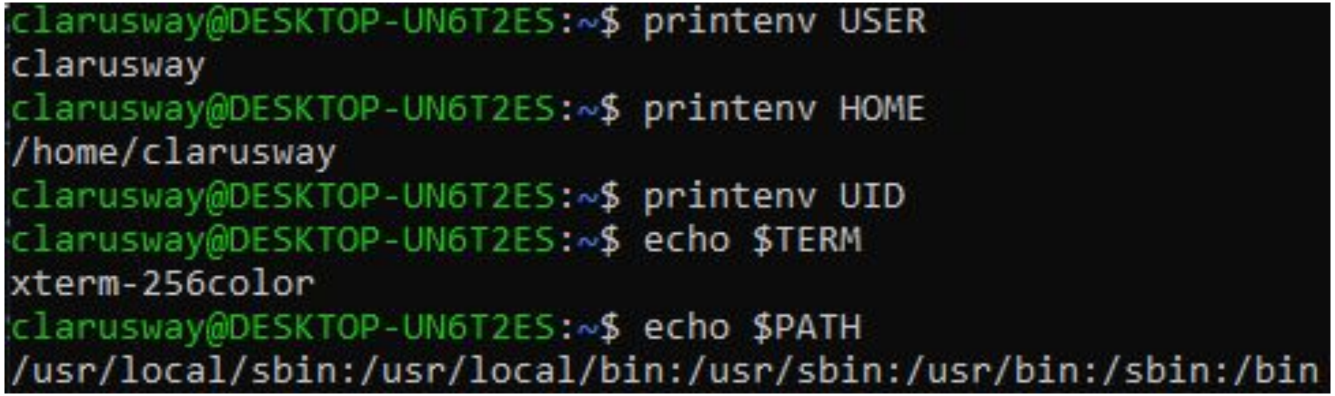
Common Commands

Command	Description
echo $VARIABLE	To display value of a variable
env	Displays all environment variables
VARIABLE_NAME=variable_value	Create a new shell variable
unset	Remove a variable
export Variable=value	To set value of an environment variable

Accessing Variable

printenv or echo

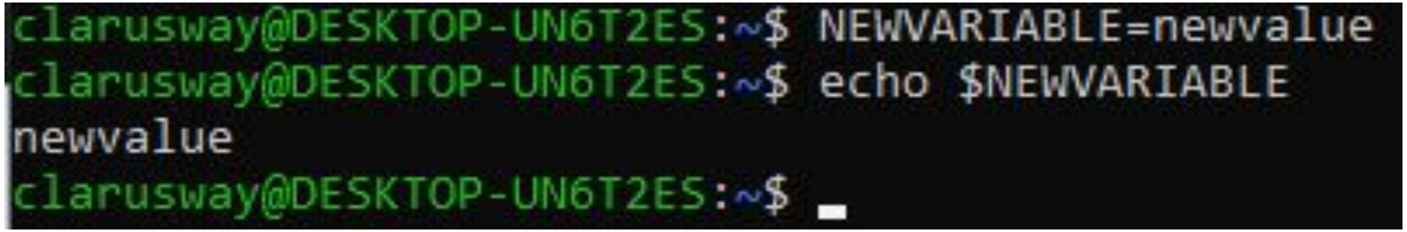
Display Path Environment Variable.



```
clarusway@DESKTOP-UN6T2ES:~$ printenv USER
clarusway
clarusway@DESKTOP-UN6T2ES:~$ printenv HOME
/home/clarusway
clarusway@DESKTOP-UN6T2ES:~$ printenv UID
clarusway@DESKTOP-UN6T2ES:~$ echo $TERM
xterm-256color
clarusway@DESKTOP-UN6T2ES:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Define a New Variable

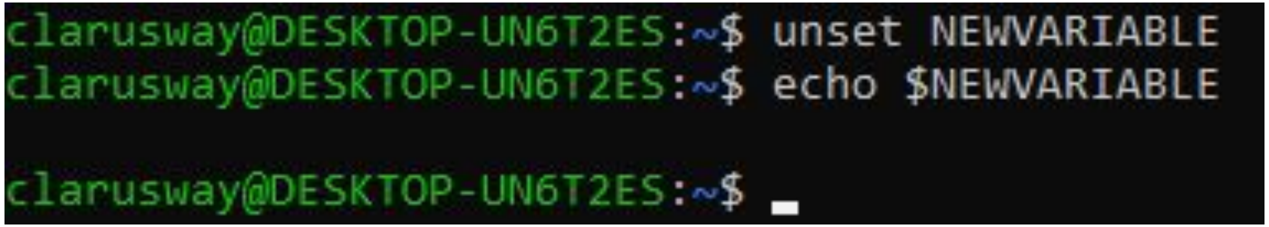
Define a new variable



```
clarusway@DESKTOP-UN6T2ES:~$ NEWVARIABLE=newvalue
clarusway@DESKTOP-UN6T2ES:~$ echo $NEWVARIABLE
newvalue
clarusway@DESKTOP-UN6T2ES:~$ _
```

Remove a Variable

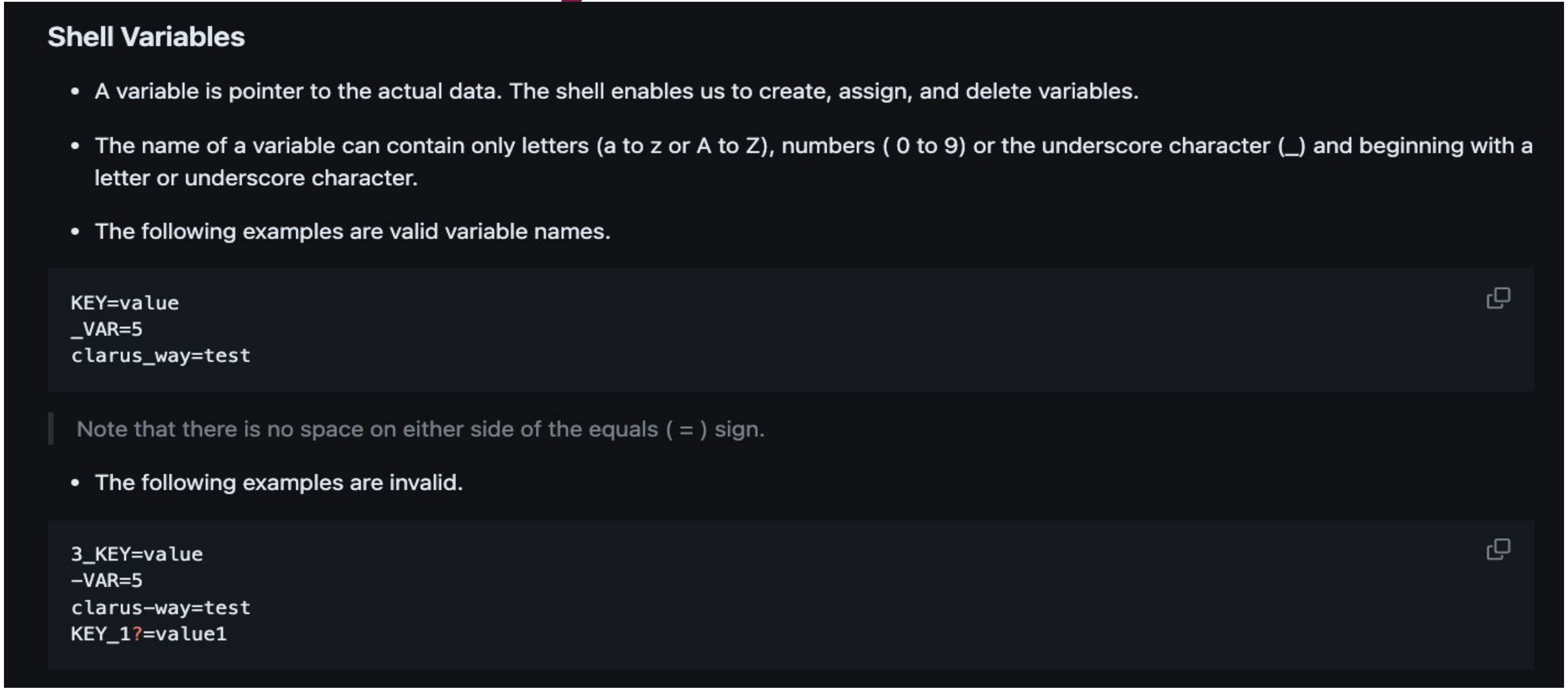
unset Remove a variable from the system.



```
clarusway@DESKTOP-UN6T2ES:~$ unset NEWVARIABLE
clarusway@DESKTOP-UN6T2ES:~$ echo $NEWVARIABLE

clarusway@DESKTOP-UN6T2ES:~$ _
```

Variable Syntax



Shell 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 following examples are valid variable names.

```
KEY=value
_VAR=5
clarus_way=test
```

> Note that there is no space on either side of the equals (=) sign.

- The following examples are invalid.

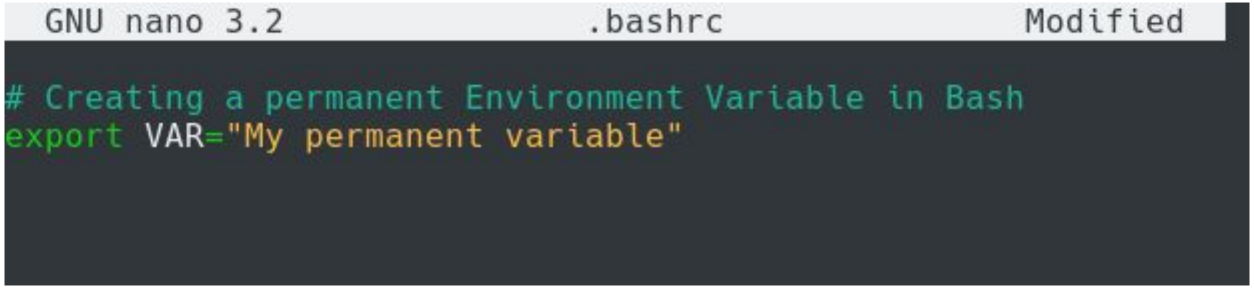
```
3_KEY=value
-VAR=5
clarus-way=test
KEY_1?=value1
```

Exercise

Create a variable named **MYVAR** with the value of **“my value”**
Print value of the **MYVAR** variable to the screen
Assign **“new value”** to the **MYVAR** variable
Print value of the **MYVAR** variable to the screen
Delete **MYVAR** variable
Print value of the **MYVAR** variable to the screen

Note

- The `export` command has the effect in the one terminal session you run it only.
- So you need to **`touch ~/.bashrc`** if you are using bash, and put your export entry in that file.
- Then to apply it in that session, do **`source ~/.bashrc`**.
- For future terminal sessions, this file will be loaded automatically.



```
  GNU nano 3.2                    .bashrc                       Modified

# Creating a permanent Environment Variable in Bash
export VAR="My permanent variable"
```

4

The PATH Variable

The PATH Variable

When we want the system to execute a command, we almost **never need to give the full path** to that command.

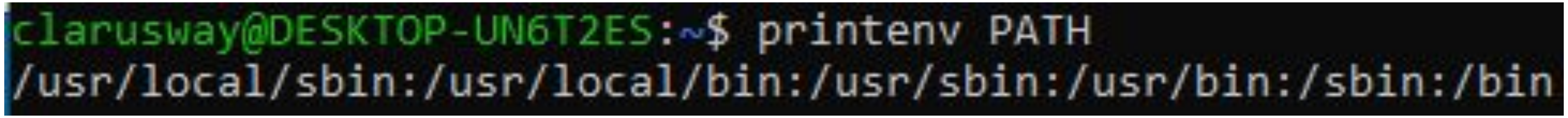
For instance, we know that the ls command is in the /bin directory (you can check with), yet we don't need to enter the /bin/ls command for the computer to list the content of the current directory.

This is maintained by the PATH environment variable. This variable **lists all directories** in the system **where executable files** can be **found**.

The PATH Variable

printenv

Display Path Environment Variable.



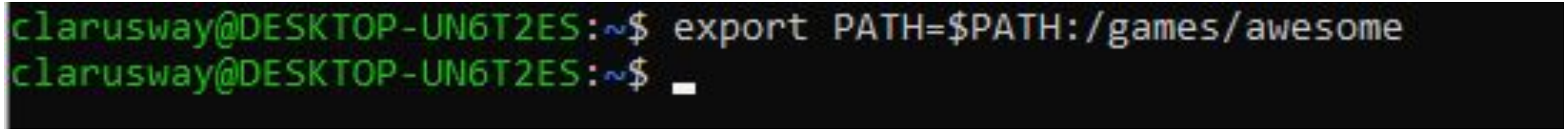
```
clarusway@DESKTOP-UN6T2ES:~$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

In this example, the /usr/local/sbin, /usr/local/bin, /usr/sbin, /usr/bin, /sbin and /bin directories are subsequently searched for the required program. The search will be stopped as soon as a match is found, even if not all directories in the path have been searched.

The PATH Variable

export

Add a New Directory to the Path.



```
clarusway@DESKTOP-UN6T2ES:~$ export PATH=$PATH:/games/awesome
clarusway@DESKTOP-UN6T2ES:~$ _
```

Let's say you want to run that file called fun. You learned from running the find command that it's in a directory called /games/awesome. However, /games/awesome is not in your path, and you don't want to type the full path just to run the game. So you can add it to PATH variable with export command.

5

Quoting with Variables

Quoting

Quoting is used to disable special treatment of certain characters and words, as well as to prevent parameter expansion and preserve what is quoted.

The bash shell knows rare, important characters. For example, $var is used to extend the value of the element.

```
echo "$PATH"
echo "$PS1"
```

CLARUSWAY
WAY TO REINVENT YOURSELF

Quoting

Double Quotes

- The double quote ("quote") protects everything enclosed between two double quote marks except $, ', " and \.

echo "$SHELL"
echo "/etc/*.conf"

Single Quotes

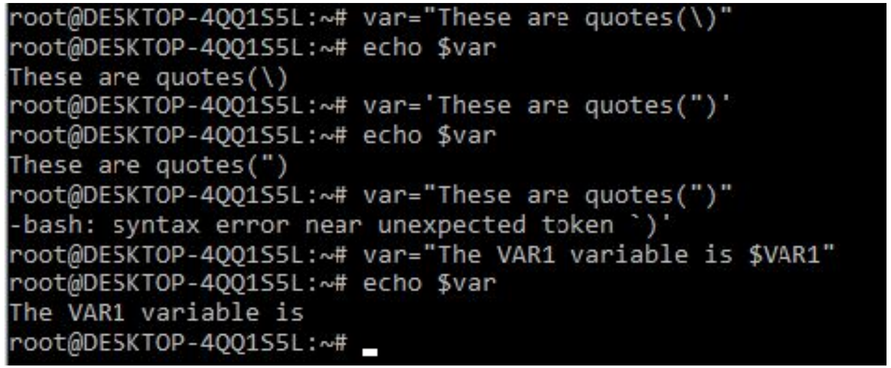
- The main point of using single quotes in Bash (') is to preserve the literal value of each character within the quotes. Single quotes strip any special value of a character and should be used when you want the input to retain its literal value without any data interpolation.

echo '$SHELL'
echo '/etc/*.conf'

Backslash

- Use the backslash to change the special meaning of the characters or to escape special characters within the text such as quotation marks.

echo "Path is \$PATH"



```
root@DESKTOP-4QQ1S5L:~# var="These are quotes(\)"
root@DESKTOP-4QQ1S5L:~# echo $var
These are quotes(\)
root@DESKTOP-4QQ1S5L:~# var='These are quotes(")'
root@DESKTOP-4QQ1S5L:~# echo $var
These are quotes(")
root@DESKTOP-4QQ1S5L:~# var="These are quotes(")"
-bash: syntax error near unexpected token `)'
root@DESKTOP-4QQ1S5L:~# var="The VAR1 variable is $VAR1"
root@DESKTOP-4QQ1S5L:~# echo $var
The VAR1 variable is
root@DESKTOP-4QQ1S5L:~# _
```

CLARUSWAY
WAY TO REINVENT YOURSELF

6

sudo Command

CLARUSWAY
WAY TO REINVENT YOURSELF

sudo Command

The sudo (**superuser do**) command gives **some admin privileges** to non-admin users.

When you put sudo in front of any command in terminal, that command **runs with elevated privileges**.

If you're not sure whether you're using sudo or su, look at the trailing character on the command line. If it's a pound sign (#), you're logged in as root.

sudo Command

Commands	Meaning
sudo -l	List available commands.
sudo command	Run command as root.
sudo -u root command	Run command as root.
sudo -u user command	Run command as user.
sudo su	Switch to the superuser account.
sudo su -	Switch to the superuser account with root's environment.
sudo su - username	Switch to the username's account with the username's environment.
sudo -s	Start a shell as root
sudo -u root -s	Same as above.
sudo -u user -s	Start a shell as user.

THANKS!

Any questions?

You can find me at:

- @sumod
- sumod@clarusway.com