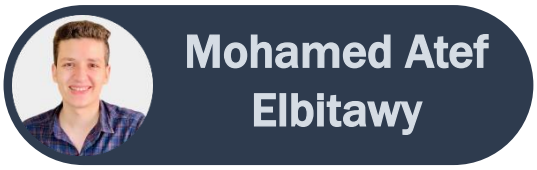
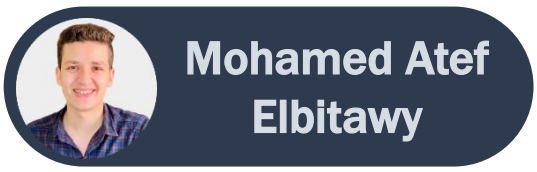


Shell Scripting

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy

https://github.com/MohamedAtefElbitawy



What is the shell scripts?

- الاسكريبت في اللينكس عباره عن text file بيبقي فيه مجموعه من ال commands وال instructions اللي بيتعملها execute باستخدام ال shell . وده بيسمحلك انك ت automate اي task عندك وده طبعا بي save ال effort وال time وبيقلل جداا الhumane mistakes اللي بتحصل incase انك بت run ال commands دي بطريقه manual

Why shell scripting?

- اول حاجه ال knowledge في ال shell scripting مهمه جداجدا و essential لأي سيستم ادمن
- تاني حاجه ايام red hat 7 كان السيستم بتاعنا وهو بي boot up كان بيروح ي execute مجموعه من ال shell script اللي بتبقي استورد تحت /etc/rc.d زي مثلا انه بيروح يعمل start up ل services زي ال networking بس ده طبعا replaced بال systemd services في red hat 8 و 9 ان ال systemd هي دلوقتي اللي بت handle ال start up بتاعت ال services
- وكمان ال syntax بتاعه سهل جدا في كتابته وكمان very straightforward

When not to use shell scripts?

- مع ان ال shell scripting is very powerful tool لينا كسيستم ادمن الا ان فيه بعض ال scenarios او ال use cases ال Shell Scripting مبيكونش هو افضل اختياريعني علي سبيل المثال في حاله ال resource intensive application ال shell script مبيكونش افضل اختيار بالاخص ان ال shell script بيبقي interpreted at runtime وده ممكن يسببلك overhead علي ال system بتاعك ف بيبقي الافضل في الحاله دي انك تستخدم compiled language زي البايثون وال C وال C++ وال Java علشان تديك best performance
- وكمان لو عايز تعمل advanced math operations او complex data processing ف الافضل في الحاله دي انك تستخدم more complex programming language
- كمان لو انت عايز تعمل application بي Generate GUI ف ال Shell Scripting هو مصمم علشان يتعامل مع ال Command Line Interface ف في حاله ال GUI بتبقي محتاج انك تدور علي programming language تانيه تستخدمها

Useful tips for writing effective bash scripts in Linux

- خلينا اقولك علي بعض ال Tips المهمه واللي هتساعدك انك تكتب effective shell scripts in Linux

- اول حاجه حاول علي قد ماتقدر انك ت include comments in your script علشان ت explain ال purpose بتاع اي section وتخلي الاسكريبت بتاعك more readable لاي حد وده بيساعد في ال troubleshooting لو في اي مشكله
- تاني حاجه ال error-handling حاول دايما تخلي الاسكريبت بتاعك ي smart enough ان هو ي detect لو فيه اي failure او اي issue ان هو ي terminate علي طول عن طريق انك تحط مجموعه من ال IF Conditions وهكذا
- تالت حاجه مهم جدا وانت بتستخدم ال Variables في الاسكريبت بتاعك حاول علي قد ماتقدر تخلي ال variables names بتاعتك more descriptive و حاول تستخدم ال double quotes علي قد ماتقدر علشان لو فيه اي special characters في ال variable بتاعك ميحصلش ليك اي unexpected errors
- اخر حاجه حاول تستخدم debugging tools علشان ت test الاسكريبت بتاعك وتتاكد انه شغال As expected

How to Write a Script.

- خلينا نشوف ازاي ن create simple shell script

1- اول خطوه هنعمل create ل file باستخدام touch command ويكون امتداد ال file ده .sh زي مثلا انا هعمل create ل file اسمه project ف هيكون عباره عن project.sh

2- بعد كده هنفتح ال file بأي text editor موجود عندك سواء كان vim او nano او gedit

3- بعد كده اول line هتحطه في ال file بتاعك حاجه اسمها sha-bang(#!) بمعني ان انت اول line هتحطه في ال file هتحط الهاش # وبعد كده علامه التعجب ! وبعد كده لازم تحددله ال path الخاص بال command interpreter اللي هيعمل execute للاسكريبت ده وده بيبقي علي حسب انت كاتب الاسكريبت بتاعك بانهي language وليكن انا هكتب الاسكريبت بالباش يبقي اول Line عندي في ال file هيكون عباره عن #!/bin/bash ودي امثله عن بعض ال sha-bang مع لغات برمجه مختلفه

```
#!/bin/sh
#!/bin/bash
#!/usr/bin/csh
#!/usr/bin/perl
#!/usr/bin/python
```

4- خلي بالك لو انت نسيت تحط اول Line في ال file هو by default هيعتبر ان انت عايز تنفذ الاسكريبت بتاعك بال bash

5- بعد كده هنعمل execute للاسكريبت علشان اعمل execute للاسكريبت عندي كذا Options

- اول options لو انا كاتب الاسكريبت بتاعي بالباش اسكريبت بستخدم command اسمه bash وبعدين اديله اسم الاسكريبت بتاعي هيروح يعمله execute بالباش **bash scriptname** . او لو انا كاتب الاسكريبت بتاعي بال sh عندي command اسمه sh وبعدين اسم الاسكريبت **sh scriptname**
- تاني options ان انا ادي execute permission للاسكريبت بتاعي عن طريق **chmod command** وليكن اقوله مثلا **chmod 555 scriptname** او ممكن استخدم الطريقه ال symbolic عن طريق ان انا اقوله **chmod +rx scriptname**

6- اخر خطوه علشان انفذ الاسكريبت ده لازم احددله ال Path الخاص بالاسكريبت عندي كذا طريقه

- اول طريقه عن طريق ان انا ممكن اقوله دوت وبعدين forward slash وبعدين اسم الاسكريبت بتاعي زي مثلا **./scriptname** بمعني لما انا قولتله دوت. معناها ان انا واقف في نفس المكان اللي فيه الاسكريبت ان ال current working directory هو نفس ال location اللي فيه الاسكريبت فهيعملي execute للاسكريبت
- تاني طريقه لو انا واقف في اي مكان تاني فساعتها لازم اديله ال full path بتاع الاسكريبت
- تالت طريقه لو انا عايز انفذ الاسكريبت بتاعي بطريقه افضل واحسن كانه command موجود عندي بمعني ان انا ممكن انفذ الاسكريبت بتاعي من اي مكان بمجرد ان انا اكتب اسم الاسكريبت فقط هيروح ينفذ الاسكريبت ده . ف علشان اعمل كده لازم احط الاسكريبت بتاعي في اي location من ال Locations بتاعت ال Path Variable زي ال location ده **/usr/local/bin** ال Path Variable بيبقي stored جواه ال locations اللي لما تكتب اي command بيروح هو يدور علي اي executable file في ال locations دي ولو لاقي اي executable file بالاسم اللي انت كاتبهوله بيروح يعمله execute علي طول

- تعال بقي نعمل create ل shell script ونكتب جواه ال command ده echo “ Hello, World!”
- اول step هعمل create ل file اسمه مثلا myscript.sh ومتنساش يكون الامتداد .sh

```
mohamed@MohamedAtef:~$ touch myscript.sh
```

- تاني step ان انا افتح ال file باي editor عندي وليكن هفتح ال file بال vim

```
mohamed@MohamedAtef:~$ vim myscript.sh
```

- تالت step بعد اما افتح ال file هكتب في اول Line ال sha-bang(#!) وبعدين احددله ال path الخاص بال command interpreter ولو انت مش عارف ال path ممكن تستخدم ال command ده which bash بعد اما تنفذ ال command ده هيقولك ان ال path بتاع ال command هو /usr/bin/bash او /bin/bash

```
mohamed@MohamedAtef:~$ which bash
/usr/bin/bash
```

- هفتح ال file بعد اما عرفنا ال path وهكتب في اول سطر #!/usr/bin/bash وبعد كده اكتب ال command اللي انا عايز انفذه وليكن انا عايز الاسكريبت ده يطبعلي Hello, Word! ف انا هقوله echo “Hello, World!”

```
#!/usr/bin/bash
echo "Hello, World!"
```

- ال step اللي بعد كده ان انا اعمل execute permissions ف علشان اعمل execute permissions هقوله chmod +x وليكن ان انا عايز ادي لكل الناس execute permissions علي الاسكريبت ده

```
mohamed@MohamedAtef:~$ chmod +x myscript.sh
```

- انا كده جاهز ان اعمل execute للاسكربت ف انا هروح اقف في نفس المكان اللي فيه الاسكربت و هقوله ./myscript.sh.

```
mohamed@MohamedAtef:~$ ./myscript.sh
Hello, World!
```

هنلاقيه هنا عملي execute للاسكريبت وقالي Hello, World!

- طب لو انا عايز اعمل execute للاسكربت من اي مكان عندي بمجرد ان انا اكتب اسم الاسكربت واضغط enter الاسكريبت ده يتنفذ عن طريق ان انا احط الاسكريبت بتاعي في اي location من ال Locations بتاعت ال Path Variable انا عندي Variable اسمه $PATH ده عباره عن variable موجود عندي في السيستم بيبقي stored جواه مجموعه من ال locations ف انت لما تكتب في ال CLI اي commandف بيروح يدور علي اي executable file بنفس ال name اللي انت كاتبهوله ويعمله execute

```
mohamed@MohamedAtef:~$ echo $PATH
/home/mohamed/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

دي كده كل ال locations اللي هيدور فيها علي ال executable file

- عن طريق ان انا هعمل move للاسكريبت في اي Location من ال Locations الموجوده عندي وليكن مثلا هعمل move للاسكريبت تحت ال location ده /usr/local/bin

```
mohamed@MohamedAtef:~$ mv myscript.sh /usr/local/bin
```

- لو جينا ننفذ الاسكريبت من اي مكان عندي علي السيستم هيتنفذ معانا بمجرد ان انا كتبت اسم الاسكريبت فقط

```
mohamed@MohamedAtef:~$ myscript.sh
Hello, World!
```

Quoting Special Characters

- هنتكلم عن quoting special characters مبدئيا انا عندي في اللينكس مجموعه من ال special characters بيبقي ليها special meaning عند ال shell بمعني ان ال shell اول اما تشوف ال characters دي بتتعامل معاهم بال special meaning بتاعهم مش بال literal meaning بتاعهم زي مثلا ال space ليها special meaning عند ال shell معناها word separated عندي مثلا **dollar sign($)** ليها special meaning وهكذا
- لو انا عايز اتعامل مع ال special characters بالمعني الحرفي او بال literal meaning بتاعهم فساعتها في الحاله دي مهم جدا انك تستخدم ال Quotations او ال Quoting
- **انا عندي 3 انواع من ال Quoting**

1- اول نوع ال **Backslash (\)** بنستخدمه في حاله لو احنا عايزين نignore ال special meaning بتاع Only one single character

2- تاني نوع ال **Single Quotation (‘)** ان اي special characters with in single Quotation بتاعتك ال bash هتتعامل معاهم بالمعني الحرفي وده بيتقال عليه full quoting لو انت عايز ت ignore ال special meaning بتاع اي special characters موجود عندك في اللينكس حطه مابين Single Quotation

3- تالت نوع ال **Double Quotation (“”)** ده ال most common use في ال scripting عموما وال double Quotation بي ignore ال special meaning بتاع بعض ال characters

- **هنشوف امثله علي ال 3 انواع علشان الصوره تكون واضحه جدا**

- يعني مثلا لو انا نفذت ال command ده echo # Hello World# مش هيظهرلي ال message دي Hello World ليه لان الباش اول اما لاقت echo فهمت ان انت عايز تعمل execute لل command ده بعد كده لما لاقت الهاش# فهمت ان انت عايز ت ignore ال command ده علشان الهاش ليها special meaning عند ال shell ان اي characters بتيجي بعد الهاش بيتعملها ignore هتتعامل معاها كانها comment كانه مش موجوده ف مظهرش اي message

```
mohamed@MohamedAtef:~$ echo # Hello World #
```

- طب لو انا عايز اعرض ال message وعايز الهاش اللي في الاول يتعملها echo وعايز الباش يتعامل معاها بالمعني الحرفي بتاعها فيه عندي كذا طريقه اول طريقه ان انت ممكن تحط قبل الهاش backslash ايه اللي هيحصل لو حطيت backslash قبل ال special characters بتاعك قالك في الحاله دي الباش اول اما تلاقي backslash هتفهم ان انت عايز تعمل ignore لل special meaning بتاع ال character اللي بعد ال backslash ف هتتعامل معاه بالمعني الحرفي ف ال message اللي هتظهر هي # Hello World

```
mohamed@MohamedAtef:~$ echo \# Hello World #
# Hello World
```

- طب لو انا عايز ignore اول هاش وتاني هاش يبقي نحط قبل اول هاش وتاني هاش backslash

```
mohamed@MohamedAtef:~$ echo \# Hello World \#
# Hello World #
```

- او استخدم تاني طريقه اني استخدم ال single Quotation انك تحط ال characters بتاعتك كلها مابين ال single Quotation وقولنا ان ال single Quotation عباره عن full quoting بيعمل disable لل special meaning بتاع اي characters جوه ال single Quotation

```
mohamed@MohamedAtef:~$ echo '# Hello World #'
# Hello World #
```

- او استخدم ال Double Quotation

```
mohamed@MohamedAtef:~$ echo "# Hello World #"
# Hello World #
```

- احنا قولنا ان ال Double Quotation بيعمل Duoting لبعض ال special characters وفيه بعض الحاجات زي ال Command Substitution وزي ال Variable Substitution بيسمحلك بيها مش بيعملها disable . تعال نشوف مثال علي النقطه دي انا عندي command عباره عن shell variable اسمه $USER وده بيكون موجود فيه ال username اللي انت عامل بيه login علي ال machine انا لو نفذت ال command ده مع ال echo هيعرضلي اسم اليوزر اللي عامل Login بيه علي ال machine زي مظاهر قدامنا اسمه mohamed

```
mohamed@MohamedAtef:~$ echo $USER
mohamed
```

- طب لو انا استخدمت معاها ال Double Quotation هل هيعرضلي ال username ولا يعمل Ignore هيعرضلي ال username لان احنا قولنا انه بيعمل quoting لبعض ال special characters وفيه بعض الحاجات زي ال Command Substitution وزي ال Variable Substitution بيسمحلك بيها

```
mohamed@MohamedAtef:~$ echo "$USER"
mohamed
```

- ولو استخدمنا نفس المثال بس مع ال Single Quotationهيعمل ignore لل dollar sign($) وهيعرضلي نفس ال command

```
mohamed@MohamedAtef:~$ echo '$USER'
$USER
```

- مهم جدا انت عندك ال Quoting سواء كان ال Single Quotationاو ال Double Quotationاو ال backslash مهمه جداا جدا مع ال File Name Handling وال Path Name Handling وبالاخص ال File Names وال Path Names اللي فيها ال Multiple Spaces. وليكن مثلا انا عايز اعمل Create ل Directory بيكون فيه Space زي مثلا Mahara Tech مابين Mahara و Tech فيه Space انا عايز اعمل Directory بالاسم ده وفيه مابينهم Space علشان اعمل كده لازم استخدم ال Quoting ولو انا مستخدمتش ال Quoting هيفهم منك انت عايز تعمل Create ل Directory باسم Mahara و Directory تاني باسم Tech زي ماهو واضح

```
mohamed@MohamedAtef:~$ mkdir Mahara Tech
mohamed@MohamedAtef:~$ ls
Tech    Mahara
```

لما عملنا Create ل Mahara Tech وعملنا List علشان نشوف ال Directory هتلاقيه عمل Create ل Two Directory واحد بأسم Mahara والتاني بأسم Tech

- ف علشان نعمل Create ل Directory واحد باسم Mahara Tech فيه عندي اكتر من طريقه ي اما استخدم ال Double Quotation عن طريق اني احط مابين ال Double Quotationال Directory الليي عايز اعمله Create

```
mohamed@MohamedAtef:~$ mkdir "Mahara Tech"
```

- ي اما ال Single Quotationبنفس طريق ال Double Quotation

```
mohamed@MohamedAtef:~$ mkdir 'Mahara Tech'
```

- او ال Backslash عن طريق ان احط ال \ backslash قبل ال Space بالشكل ده Mahara\ Tech

```
mohamed@MohamedAtef:~$ mkdir  Mahara\ Tech
```

- لو عملنا list وشوفنا ال Directory بعد اما اتعمله Create هنلاقي ال Directory مابين Single Quotation علشان يوضحلك ان ده عباره عن Directory واحد اسمه Mahara Tech

```
mohamed@MohamedAtef:~$ ls
 'Mahara Tech'
```

- مش recommended ان انت تعمل Directory يكون فيه Spaces علشان ده ممكن يعملك مشاكل بعد كده بمعني ان احنا لو عايزين ن Access ال Directory ده وندخل جواه وجينا واحنا بن access ال Directory عملنا cd Mahara Tech هيطلعلك error ويقولك too many arguments

```
mohamed@MohamedAtef:~$ cd Mahara Tech
bash: cd: too many arguments
```

- ف مهم جدا وانت بت Access اي Directory فيه Space لازم ال Directory ده مابين Double Quotation

```
mohamed@MohamedAtef:~$ cd "Mahara Tech"
mohamed@MohamedAtef:~Mahara Tech$
```

- او ممكن ان احنا نستخدم ال Backslash عن طريق ان ان قبل ال Space احط backslash

```
mohamed@MohamedAtef:~$ cd Mahara\ Tech
mohamed@MohamedAtef:~Mahara Tech$
```

- لو احنا عرفنا Two Variable ال x = 5 وال y = 10 وعملنا وعرضنا قيمه ال Two Variables باستخدام echo $x $y هتلاقيه قايلك ان ال x ب 5 وال y ب 10

```
mohamed@MohamedAtef:~$ x=5
mohamed@MohamedAtef:~$ y=10
mohamed@MohamedAtef:~$ echo $x $y
5 10
```

Special Characters

- انا عندي بعض من ال Special Characters بيبقي ليهم Special Meaning علي حسب انت بتستخدمهم مع انهو Command

dot (.)

- ال **dot .** وده بيكون ليه كذا Special Meaning

- اول Special Meaning لو انت بتستخدمها مع Touch Command وحطيت dot في بدايه ال File Name كده ال Bash هيفهم منك انك عايز تعمل Create ل Hidden File

```
[root@server~]# touch .hidden File
```

- تاني Special Meaning لو انت بتستخدممع ال cd command ال dot هنا بتعبر عن ال Current Working Directory بتاعك يعني مثلا لو ال Current Working Directory بتاعك كان /home/mohamed/project وقولتله **cd ..** هيفهم منك ان انت عايز ترجع لل Parent Directory بتاعك اللي هو /home/mohamed

```
[root@server~]# pwd
/home/mohamed/projects
[root@server~]# cd ..
[root@server~]# pwd
 /home/mohamed
```

- تالت Special Meaning لو استخدمنا ال dot مع ال Copy مع ال cp command ان انا مثلا عايز اعمل copy من ملف في المكان اللي انا فيه زي مثلا انا عايز اخد copy من /home/mohamed/projects/ ل المكان اللي انا فيه ف بيكون عندي طريقتين ي اما اقوله dot ي اما احددله ال Path اللي انا عايز اعمل Copy ليه

```
[root@server~]# cp /home/mohamed/projects/ .
```

Backquotes or backticks `

- ال `Backquotes معناها Command Substitutions ان انت لو عايز ت execute command معين وال Value بتاعتك بدل اما تعملها Display علي الاسكرين تعملها assign او

```
[root@server~]# DATE=`date`
[root@server~]# echo $DATE
The Dec 24 21:35:27 EET 2024
```

تعملها Store ل variable معين فعلشان تعمل هتقوله مثلا ال Variable بتاعك وليكن DATE بيساوي = ومابين Backquotes هتحط ال Command اللي انت عايز تعمله executed زي مثلا `date`

Exclamation Mark!

- علامه التعجب ! عندها Special Meaning في ال Shell ومعناها ال invert وبتستخدم مع ال Test Operator او مع ال If Condition وبنستخدمها كتير جدا مع ال Bash Script

Asterisk *

- ال Asterisk ودي بتستخدم مع ال Regular Expression وبت represents zero or more characters وبتستخدم مع ال Arithmetic Operations او ال Multiplication

Question Mark ?

- ال Question Mark بتستخدم مع ال Regular Expression وبت Match علي one single characters شبه ال dot بالظبط بس الفرق بينها ومابين ال dot ان ال dot بتستخدم مع ال grep command اما ال Question Mark بتستخدم مع ls command

Dollar Sign $

- انا عندي ال **$** Dollar Sign ليها اكتر من Special Meaning اول Special Meaning بتستخدم مع ال Variable Substitutions يعني مثلا لو قولتله echo $x هو كده هيعمل Display لل Content بتاع ال Variable اللي اسمه x
- تاني Special Meaning بتستخدم في حاله ال Regular Expression بالاخص مع ال grep command لو انا عايز اعمل Search علي اي Line بينتهي ب Pattern معين

$?

- لو قولتله **$?** ف ال Question Mark? عباره عن Variable بيبقي Stored جواها حاجه اسمها ال Exit Code او ال Exit Status بتاعت اخر Command انت عملتله Execute سواء كان من ال CLI او في الاسكريبت بتاعك وال Question Mark بيبقي Stored جواها ي اما Zero ي اما Non Zero . و Zero معناها Successful ان ال Command اللي انت عملتله Execute بيكون Successful اتنفذ بشكل صحيح . و Non Zero معناها Not Successful

$$

- لو قولتله **$$**كده هيعمل Display لل Process ID بتاعت ال Current Bash بتاعتك . لو انت جوه الاسكريبت بتاعك وقولتله $$ echo هيعمل Display لل Process ID بتاع الاسكريبت بتاعك

Parentheses()

- بتستخدم في حاله ال Command Grouping زي مثلا (a=hello;echo $a)
- وبتستخدم في حاله ال Array Initialization زي مثلا

Array = (element1 element2 element3)

Brace Expansion {xxx,yyy,zzz}

- بستخدمها لو انا عايز اعمل Create ل اكتر من file مره واحده زي مثلا

touch {file1,file2,file3,file4}

```
[root@server~]# touch {file1,file2,file3,file4}
[root@server~]#ls
file1    file2    file3    file4
```

Extended Brace Expansion {a..z}

- ال Extended Brace Expansion ليها Special Meaning ودي معناها ان انت عايز تطبع Range معين مثلا لو قولتله echo {a..z} ده كده هيفهم منك انت عايز تطبع ال range من a لحد z وهيطبعلك كل الحروف بتاعتك. ولو قولتله echo {1..20} هيطبعلك كل الارقام من 1 لحد 20

```
[root@server~]# echo {a..z}
a b c d e f g h i j k l m n o p q r s t u v w x y z
[root@server~]# echo {1..20}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

Pathname {} \;

- بنستخدمها اكتر مع ال Find Command ان انا بستخدمه مع ال find علشان اعمل Search علي حاجه معينه وليكن مثلا عايز اعمل Search علي file معين ف ممكن اقوله find في ال Current Working Directory ان اي File ال Name بتاعه اسمه file1 روح اعمله execute ل rm -f وامسح ال file ده وطالما استخدمت ال exec لازم تحطله ال Path name بالشكل ده {} \;

```
[root@server~]# find . -name file1 -exec rm -f {} \;
```

Square Brackets [] and Double Square Brackets [[ ]]

- لو انا عايز اعمل Condition معين او Test Operator معين بنستخدم ي اما square brackets [ ] ي اما [[ ]] Double square brackets الاتنين شبه بعض بس تقدر تقول ان ان ال [[ ]] بتكون More Flexible عن ال [ ]

Redirection > &> >& >> <

- ال Redirection عباره عن feature في اللينكس بستخدمها علشان خاطر أredirect ال output بتاع اي command وأ save ال output ده في file معين

➢ لو انا نفذت ال command ده اللي هو **ls -l** ف طبيعي انه هيعرضلي ال output علي الاسكرين او ال Terminal زي مظاهر كده

```
mohamed@MohamedAtef:~$ ls -l
-rw-------. 1 root root 1380 Mar  5 19:43 anaconda-ks.cfg
drwxr-xr-x. 3 root root   17 Jul  7 21:48 backup
drwxr-xr-x. 2 root root  163 Jul  8 15:30 Desktop
drwxr-xr-x. 2 root root    6 Jul  7 14:24   devops
```

➢ طب لو انا عايز اعمل redirect لل output ده بدل اما يطلعلي علي الاسكرين هعملsave لل output ده في file معين هقوله **ls -l** وبعدين علامه الاكبر من **>** وبعدين احددله ال file

```
mohamed@MohamedAtef:~$ ls -l > /tmp/temp.txt
```

انا عملت redirect لل output بتاع ls -l بدل اما يعرضه علي الاسكرين لا هيعمله save في file اسمه temp.txt تحتtmp/

➢ لو جينا دلوقتي عملنا cat لل temp.txt هنلاقيه عمل save لل output في ال file

```
mohamed@MohamedAtef:~$ cat /tmp/temp.txt
-rw-------. 1 root root 1380 Mar  5 19:43 anaconda-ks.cfg
drwxr-xr-x. 3 root root   17 Jul  7 21:48 backup
drwxr-xr-x. 2 root root  163 Jul  8 15:30 Desktop
drwxr-xr-x. 2 root root    6 Jul  7 14:24 devops
```

➢ مثال تاني فيه عندي command اسمه date ال command ده بيعرضلي التاريخ والوقت لو انا مش عايزه يعرضلي ال output علي الاسكرين هستخدم علامه اكبر من وبعدين اسم ال file

```
mohamed@MohamedAtef:~$ date > date.txt
```

➢ لو عملنا cat لل date.txt هنلاقي ال output معمول ليه save في ال file

```
mohamed@MohamedAtef:~$ cat date.txt
Wed Jul 10 02:09:30 PM EET 2024
```

- لو انا عايز اعمل append علي file معين بمعني لو انا عندي file عليه داتا وانا عايز اعمل redirect بس ميمسحش الداتا اللي علي ال file ويبدلها بالداتا الجديده هستخدم علامه اكبر من مرتين <<علشان اعمل append

```
mohamed@MohamedAtef:~$ cal >> date.txt
mohamed@MohamedAtef:~$ cat date.txt
Wed Jul 10 02:09:30 PM EET 2024
      July 2024
Su Mo Tu We Th Fr Sa
   1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

- لو احنا عملنا run ل اي Command مش موجود هيطلعلك error

```
mohamed@MohamedAtef:~$ saddasd
bash: saddasd: command not found...
Failed to search for file: /media/BaseOS/ was not found
```

- لو انا اعايز عمل save لل standard error ل file معين بدل اما يظهرلي علي الاسكرين هستخدم 2> بيعمل redirect لل output في ال file اللي انا عايز احفظ ال error ده فيه

```
mohamed@MohamedAtef:~$ saddasd 2> error.txt
```

- ولو انا عايز اعمل append علي ال Standard error هستخدم 2>>

```
mohamed@MohamedAtef:~$ saddasd 2>>  error.txt
```

pipe |

- بنستخدمها لو احنا عايزين ن Send ال Standard output الخاص ب ال First Command ك Input لل Second Command

- يعني في المثال التالي انا استخدم ال Pipeline علشان اعمل translate لكل ال Lower Case ل Upper Case

```
mohamed@MohamedAtef:~$ ls -l | tr 'a-z' 'A-Z'
-RW-RW-R-- 1 MOHAMED MOHAMED 0 06:42 15 أغس FILE1
-RW-RW-R-- 1 MOHAMED MOHAMED 0 06:42 15 أغس FILE2
-RW-RW-R-- 1 MOHAMED MOHAMED 0 06:42 15 أغس FILE3
-RW-RW-R-- 1 MOHAMED MOHAMED 0 06:42 15 أغس FILE4
```

Variable

- الـ Variables بتبقى زي متغيرات في أي لغة برمجة، الهدف منها إنها تخزن بيانات نقدر نستخدمها بعد كده في الاسكربت.
- علشان نعرف Variables بنكتب اسم المتغير وبعده علامة = وبعدها القيمة اللي عايزين نخزنها. مفيش مسافات قبل وبعد علامه اليساوي = زي مثلا name="Mohamed" او age=25 وهكذا

```
mohamed@MohamedAtef:~$ name= "Mohamed"
mohamed@MohamedAtef:~$ age=25
```

- علشان نطبع القيمه الموجوده في ال name بنحط $ قبل اسم المتغير علشان يعرف ان ده عباره عن Variable ولو عايز اطبع القيمه هستخدم echo $name

```
mohamed@MohamedAtef:~$ echo $name
Mohamed
```

- ولو انت مستخدمتش ال dollar sign $ هيطبعلك ال name كانه اسم

```
mohamed@MohamedAtef:~$ echo name
name
```

- طب لو انا استخدمت مع ال variable ال Double Quotation هل هيعرضلي ال username ولا هيعمل Ignore هيعرضلي ال Variable وهيعمل enable لان احنا قولنا انه بيعمل quoting لبعض ال special characters وفيه بعض الحاجات زي ال Command Substitution وزي ال Variable Substitution بيسمحلك بيها

```
mohamed@MohamedAtef:~$ echo "$name"
Mohamed
```

- ولو استخدمنا نفس المثال بس مع ال Single Quotationهيعمل ignore لل dollar sign($) وهيعرضلي نفس ال command

```
mohamed@MohamedAtef:~$ echo '$name'
$name
```

- لو جينا عملنا x=5 و y=6 وجينا عملنا z=x+y هل لو احنا طبعنا قيمه ال z هيطلعلنا مجموعهم ولا ايه بالظبط هيطبعلنا x+y لانه فهم ان ال x+y عباره عن String

```
mohamed@MohamedAtef:~$ x=5
mohamed@MohamedAtef:~$ y=6
mohamed@MohamedAtef:~$ z=x+y
mohamed@MohamedAtef:~$ echo $z
x+y
```

- ولو احنا عملنا z=$x+$y هيطبعلي قيمه 5+6 ك text

```
mohamed@MohamedAtef:~$ echo $z
5+6
```

- علشان نحل المشكله دي عندي اكتر من طريقه اول طريقه ان وانت بتعرف ال z لازم تعمل declare -i ال -i يعني Integer واقوله ان z=$x+$y ولو عملنا echo $z هيطلعنا قيمه ال z ب 11

```
mohamed@MohamedAtef:~$ declare -i  z=$x+$y
mohamed@MohamedAtef:~$ echo $z
11
```

- تاني طريقه ودي الافضل في الاستخدام ان انا اقوله z وبعدين يساوي وبعدين $ dollar sign وبين قوسين زي كده (( )) بضيف جواهم $x+$y

```
mohamed@MohamedAtef:~$ z=$(($x+$y))
mohamed@MohamedAtef:~$ echo $z
11
```

- تالت طريقه ان انا ممكن استخدم ال let عن طريق ان انا هقوله let z=$x+$y

```
mohamed@MohamedAtef:~$ let z=$x+$y
mohamed@MohamedAtef:~$ echo $z
11
```

- رابع طريقه ان انا ممكن استخدم ال typeset -i

```
mohamed@MohamedAtef:~$ typeset -i z=$x+$y
mohamed@MohamedAtef:~$ echo $z
11
```

- سؤال مختلف هل ال Variable دي بتكون موجوده فين او بتتحفظ فين وهل فعلا بتكون موجوده ممكن ان احنا نستخدم مثلا command اسمه set ونعمل grep علي ال Variable ونشوف هل هو موجود ولا لا
- لو جينا عرفنا variable اسمه Mohamed_Atef وخليناه بيساوي 26 وعايزين نعرف هل ال Mohamed ده موجود فعلا ولا لا ف هنستخدم ال set مع ال grep ونشوف

```
mohamed@MohamedAtef:~$ mohamed_Atef=26
mohamed@MohamedAtef:~$ set | grep "mohamed_Atef"
mohamed_Atef=26
```

- لو انا عندي Variable وعايز امسحه بعمل unset لل Variable

```
mohamed@MohamedAtef:~$ unset mohamed_Atef
```

- **Environment Variables**
- الأمر export في الشيل بيُستخدم لتحويل ال Shell Variable إلى Environment Variables، بحيث تكون متاحة ليس فقط في ال Shell الحالية، لكن كمان لأي عمليات أو برامج تشتغل في نفس ال Shell.
- بمعنى تاني، لما تعمل export ل Variable، ال Variable ده هيبقى متاح في كل العمليات الفرعية اللي تبدأ من ال Shell دي. العمليات دي ممكن تكون أسكربتات شيل أو برامج تانية.

```
mohamed@MohamedAtef:~$ export mohamed_Atef=26
mohamed@MohamedAtef:~$ echo $mohamed_Atef
26
mohamed@MohamedAtef:~$ bash
mohamed@MohamedAtef:~$ echo $mohamed_Atef
26
```

لو احنا عملنا export ل variable اسمه Mohamed_Atef وعملنا echo ليه هيطبعلنا القيمه بتاعته ولو فتحنا shell تانيه باستخدام command اسمه bash وعملنا echo لنفس ال variable هيطبعلنا ال قيمه ال variable

- ف ال export بيورث قيمه ال Variable لباقي الشيل

Testing Script Inputs

- احنا بنستخدم ال **Testing Conditions** علشان خاطر ن control ال Flow الخاص بال Program بتاعنا يعني Based علي Conditions معين لو اتحقق روح اعمل decision معين وهكذا . ومن ضمن ال Types ان انت تخلي الاسكريبت بتاعك smart enough انه ي dedicate لو فيه اي Failure و ي Terminate او ي Exit علي طول وبنعمل ده عن طريق ان احنا بنحط مجموعه من ال Test او ال IF Conditions في الكود بتاعنا و Based علي ال Result بتاعها بناخد decision او Action معين وهكذا
- فيه اكتر من طريقه ممكن استخدمها علشان اقدر اعمل Test Condition في الاسكريبت بتاعي

1- اول طريقه وهي if/then statement ودي شبه اي Programming Language ان انت بتقوله لو ال Condition اتحقق اعمل كذا else روح اعمل كذا وال Condition ده بيرجعلي ي True ي False في اللينكس True معناها 0 و False معناها 1

2- تاني طريقه وهي عن طريق ال [[ ]] Double Square Brackets او Single Square Brackets []

3- او ان انا استخدم ال Test

- **فيه انواع مختلفه من ال Tests**

1- String Testing

- بنستخدم ال String Testing لعمل Comparison مابين Two String باستخدام ال Test Command
- ال Comparison Operators اللي ممكن نستخدمها في حاله ال String Testing هي

=	Equal
!=	Not equal to
<=	Less than or equal to
>=	Greater than or equal to
<	Less than
>	Greater than
-z	string is null, that is, has zero length
-n	string is not null.

- مثال لو انت قولتله مثلا اعمل test هل ال String اللي اسمه Mohamed بيساوي ال String اللي اسمه Mohamed ولا لا ف علشان تشوف قيمه ال test لازم تشوف هل ال command اتنفذ ولا لا باستخدام echo $? ف ظهر قدامنا ب 0 ومعناها Successful ان ال Two String بيساوا بعض

```
mohamed@MohamedAtef:~$ test "Mohamed" = "Mohamed"
mohamed@MohamedAtef:~$ echo $?
0
```

- مثال في حاله ال Not equal to

```
mohamed@MohamedAtef:~$ test "Mohamed" != "Ahmed"
mohamed@MohamedAtef:~$ echo $?
0
```

- لو انت عندك Variable معين وانت عايز تتاكد هل ال Variable ده بيحتوي علي Value او لا ف بنستخدم -z او -n و -z بنستخدمه علشان نتاكد مما إذا كان ال Variable او ال String فارغة ولو ال Variable مبيحتويش علي اي Value ف هتكون قيمه ال $? ب Zero اما لو بيحتوي علي اي Value هتكون قيمه ال $? ب 1
- اما ال -n بتكون عكس ال -z

```
mohamed@MohamedAtef:~$ var="Devops"
mohamed@MohamedAtef:~$ test -z "$var"
mohamed@MohamedAtef:~$ echo $?
1
```

2- Numeric Testing

- بستخدم ال Integer Testing لو انا عايز اعمل Comparison مابين الارقام و انواع ال Numeric conditions

int1 –eq int2	equal to
int1 –ne int2	not equal to
int1 –gt int2	greater than
int1 –ge int2	greater and equal
int1 –lt int2	less than
int1 –le int2	less and equal

➢ مثال :- المثال التالي عباره عن مقارنه مابين رقمين ان هل ال 56 بتساوي ال 50 اكيد لاف كانت القيمه ب 1

```
mohamed@MohamedAtef:~$ test 56 -eq 50
mohamed@MohamedAtef:~$ echo $?
1
```

➢ مثال :- المثال التالي عباره عن مقارنه مابين رقمين ان هل ال 56 لا تساوي ال 50 ام لا

```
mohamed@MohamedAtef:~$ test 56 -nq 50
mohamed@MohamedAtef:~$ echo $?
0
```

3- File Testing

- فيه عندي اكتر من Option ممكن استخدمهم علشان اعرف نوع ال File وال Directory اللي عندي اذا كان مثلا File او Directory او regular file او Symbolic Link او ان انا اعرف هل ال File ده readable او writable

-e	Check if the file exists
-f	Check if the file is a regular file
-d	Check if the file is a directory
-r	Check if the file is readable
-w	Check if the file is writable
-x	Check if the file is executable

➢ مثال :- لو انا عايز اتاكد هل ال File ده واخد read permissions ولا لا هستخدم اوبشن -r

```
mohamed@MohamedAtef:~$ test -r file1
mohamed@MohamedAtef:~$ echo $?
0
```

➢ مثال عن ال regular file هستخدم اوبشن -f

```
mohamed@MohamedAtef:~$ test -f file1
mohamed@MohamedAtef:~$ echo $?
0
```

➢ مثال عن اذا كان ال file ده بيكون Directory او لا هستخدم اوبشن d-

```
mohamed@MohamedAtef:~$ test -d dir1
mohamed@MohamedAtef:~$ echo $?
0
```

➢ لو انا عايز اتاكد هل ال file ده موجود بالفعل ولالا هستخدم اوبشن e-

```
mohamed@MohamedAtef:~$ test -e script.sh
mohamed@MohamedAtef:~$ echo $?
0
```

Logical Operators

- بنستخدم ال Logical Operators لو احنا عايزين نعمل اكتر من Condition في نفس الوقت
- في عندي 3 انواع من ال Logical Operators اللي ممكن استخدمهم

1- AND (-a)

- بستخدم AND لو انا عايز اتحقق من ان جميع ال Conditions صحيحه ب Trueعن طريق ان انا بستخدم اوبشن a- مابين ال Conditions

```
mohamed@MohamedAtef:~$ test 50 -eq 50 -a "mohamed" = "mohamed"
mohamed@MohamedAtef:~$ echo $?
0
```

2- Not (!)

- بستخدم Not علشان اعكس نتيجه ال Condition لو كان ب True هيكون ب False ولو كان ب False هيكون ب True

```
mohamed@MohamedAtef:~$ test ! 50 -eq 50
mohamed@MohamedAtef:~$ echo $?
1
```

3- OR (-o)

- لو انا عايز اشوف لو شرط من الشرطين او اكتر كان صح هستخدم اوبشن o-
- عندي في المثال التالي فيه شرطين اول شرط هل 70 بتساوي 50 وتاني شرط هل string اللي اسمه Mohamed بيساوي Mohamed ف انا عندي شرط واحد صح والتاني غلط ف لما استخدمنا o- وعملنا echo $? رجعلنا 0 لان شرط واحد اتحقق

```
mohamed@MohamedAtef:~$ test 70 -eq 50 -o "mohamed" = "mohamed"
mohamed@MohamedAtef:~$ echo $?
0
```

➢ مثال تاني لو الشرطين متحققوش هتكون النتيجه ب 1

```
mohamed@MohamedAtef:~$ test 70 -eq 50 -o "mohamed" = "Ahmed"
mohamed@MohamedAtef:~$ echo $?
1
```

The read command

- أمر read في الشيل سكريبت بيستخدم لقراءة المدخلات من المستخدم أو من ملف، وتخزينها في متغيرات.
- read بيتعامل مع المدخلات اللي بتدخل من الـ Terminal أو ملف وبيحطها في متغيرات عشان تقدر تستخدمها بعد كده في السكريبت بتاعك

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

- شرح المثال:

- "echo "Enter your name": هنا السكريبت بيطبع رسالة للمستخدم تطلب منه يدخل اسمه.
- read name: هنا السكريبت بيستنى المستخدم يدخل حاجة، وأي حاجة المستخدم يكتبها هتتحفظ في المتغير name.
- "echo "Hello, $name": السكريبت بيطبع رسالة ترحيب باستخدام الاسم اللي المستخدم دخله

- أول ما المستخدم يكتب اسمه ويضغط على Enter، السكريبت هيخزن الاسم ده في المتغير name و بعد كده، السكريبت هيستخدم المتغير ده عشان يطبع رسالة ترحيب

➢ مثال تاني :-

```
echo "Enter your first name and last name:"
read first_name last_name
echo "Hello, $first_name $last_name!"
```

- شرح المثال:

- لو المستخدم كتب "Mohamed Atef" السكريبت هيخزن "Mohamed" في first_name و"Atef" في last_name.

- فيه Options ممكن استخدمها مع ال read وهي

-p	تستخدم لطباعة رسالة قبل القراءة
-s	بتستخدم لو عايز المدخلات تكون مخفية (مفيدة في الباسورد)
-n	بتحدد عدد الأحرف اللي تقدر تقراهم
-t	بتحدد وقت الانتظار قبل ما الـ read ينتهي لوحده

- مثال باستخدام p- . هتلاقي هنا ال read -p بيسألك عن سنك وفي نفس الوقت بيقرأ ال Inputs من المستخدم.

```
read -p "Enter your age: " age
echo "Your age is $age"
```

Conditional Structures

- فيه كذا نوع لل IF Conditions في ال Shell Script

1- اول نوع وهو ال **if-else statement** وبيكون ال Syntax بتاعها بالشكل ده

- ان انت في اول سطر بتقوله if ومابين [ ] Square Brackets بتحددله Condition معين
- في تاني سطر بتكتب then لو ال Condition صحيح اعمل execute ل Command 1 و Command 2
- وبعد كده بتكتب else وده بيكون Optional ان لو ال Condition غير صحيح اعمل execute ل Command 3 و Command 4
- واخر حاجه بتكتب fi علشان يفهم ان انت خلصت ال If Condition بتاعتك
- ف لو ال Condition بتاعك كان ب True هينفذلك ال Command الاولاني والتاني ولو ال Condition كان ب False هينفذلك ال Command التالت والرابع
- ممكن ان انت تكتب then في نفس السطر اللي فيه if للتبسيط زي **if [condition]; then**

```
if [ condition-true ]
then
   command 1
   command 2
   ...
else
   # Optional (may be left out if not needed).
   # Adds default code block executing if original condition tests false.
   command 3
   command 4
   ...
fi
```

➢ **مثال :-** هنعمل اسكريبت وهيكون عباره عن ان احنا هنعمل test لو في file اسمه script1.sh هعمله cat ان اعرض محتوي ال file ده ولو مش موجود هيطلعلي message بتقولي No file named script1.sh

```
script.sh                                                    bash
#!/bin/bash

if test -f script1.sh
then
    cat script1.sh
else
    echo No file named script1.sh
fi
```

- عند تنفذ الاسكريبت هيعرضلي محتوي ال script.sh لان الشرط اتحقق ان بالفعل فيه file اسمه script1.sh وبيحتوي علي

```
mohamed@MohamedAtef:~$ ./script.sh
#!/bin/bash
echo "Hello_World!"
```

- لو نفذنا نفس المثال ولكن ال file غير موجوده هينفذ تاني Condition وهو انه هيقولك ان ال file غير موجود

```
mohamed@MohamedAtef:~$ ./script.sh
No file named script1.sh
```

- فيه طريقه تانيه ان انا اعمل Test بيها وهي ان انا هستبدل ال test ب Single Square Brackets []

```
script.sh                                                    bash
#!/bin/bash

if [ -f script1.sh ]
then
    cat script1.sh
else
    echo No file named script1.sh
fi
```

- **مثال:-** في المثال الاتي هنستخدم أمر read مع جملة شرطية if عشان تتحقق من وجود ملف معين على السيستم

- هنا السكريبت بيطلب من المستخدم إنه يدخل اسم الملف اللي بيدور عليه ف السكريبت بيستنى المستخدم يكتب اسم الملف اللي بيدور عليه، وبيخزن الاسم اللي المستخدم دخله في متغير اسمه filename وبعدين بيجي دور جملة الشرط if. الشرط ده بيستخدم f- للتحقق إذا كان الملف اللي دخلت و موجود في المتغير filename هو ملف عادي (مش ديركتوري مثلًا) وفعلاً موجود في النظام لو الشرط اتحقق هيطبعلك This is file exit ولو الشرط متحققش ان ال file مش موجود هيطبعلك This is file does not exist

```
script.sh                                                    bash
#!/bin/bash
echo  "which file are you looking for?"
read  filename
if [ -f $filename ]
then
     echo  This is file  exist!
else
    echo  This is file does not exist
fi
```

- لما تشغل السكريبت هيطلب منك تدخل اسم الملف اللي بيدور عليه بعد ما تكتب اسم الملف وتضغط Enter، السكريبت هيشوف إذا كان الملف ده موجود ولا لأ. لو موجود هتظهر لك رسالة " This file exists!" لو مش موجود، هتظهر لك رسالة "This file does not exist".

```
mohamed@MohamedAtef:~$ ./script.sh
which file are you looking for?
File1                    # هنا هنلاقيه طلب منا ان احنا نكتب اسم الملف اللي احنا هنشوفه موجود ولا لا
This is file does not exist
```

2- تاني نوع وهو ال **if..elif..else statement** وده بنستخدمه لو احنا عندنا اكتر من Condition وبيكون ال Syntax بتاعها بالشكل ده . ان لو ال Condition الاولاني كان ب True اعمل Execute لل Command الاولاني والتاني والتالت وبعد كده بتقوله elif ودي معناها else if وبعد ال elif بتقوله then زي ال if بالظبط ان لو ال Condition التاني كان ب True اعمل execute لل Command الرابع والخامس ولو ال Condition الاولاني كان ب False وال Condition التاني كان ب false ساعتها هيروح ي execute ال default-command الموجود بعد ال else

```
if  [ condition1 ]
then
   command1
   command2
   command3
elif  [ condition2 ]
then
   command4
   command5
else
   default-command
fi
```

- في المثال التالي مستخدم شرطين اول شرط هيروح يتحقق لو ال filename اللي انا هدخلهوله بيكون Directory هيطبع this is a Dir! وتاني شرط لو ال filename بيكون file هيطبع this is a file!

script.sh bash

```
#!/bin/bash
echo  "which file are you looking for?"
read  filename
if [ -d $filename ]
then
      echo  " this is a Dir! "
elif [ -f $filename ]
then
     echo  " this is a file! "
else
     echo  " Invalid input !! "
fi
```

- لما تشغل السكريبت هيطلب منك تدخل اسم الfilename بعد ما تكتب اسم filename وتضغط Enter، السكريبت هيشوف إذا كان ال filename ده file ولا Directory لو Directory هيعرضلك this is a Dir ولو file هيعرضلك this is a file غير كده هيعرضلك Invalid input

```
if [ condition1 ]
then
   if [ condition2 ]
   then
      do-something # But only if both "condition1" and "condition2" valid.
   fi
fi
```

3- تالت نوع وهو ال **Nested if/ then Condition** ودي معناها if condition جوه ال if condition الاساسيه ان لو ال condition الاولاني كان ب True ادخل علي ال if condition التانيه

```
mohamed@MohamedAtef:~$ ./script.sh
which file are you looking for?
script1.sh
this is a file!
```

The Case Command

- أمر case في الـ shell scripting بيبقى زي الـ switch في لغات برمجة تانية. الهدف منه هو تنفيذ أوامر معينة بناءً على القيم المختلفة لمدخل معين. بمعني بنستخدمه لما يبقى عندك input معين وعايز تشوف قيمته إيه وتنفذ أوامر معينة بناءً على القيمة دي. وبيكون ال Syntax بتاعه بالشكل ده

```
case variable in
  value1)
    # commands to execute if variable matches value1
    ;;
  value2)
    # commands to execute if variable matches value2
    ;;
  *)
    # commands to execute if no values match (default case)
    ;;
esac
```

➢ في المثال التالي بيطلب من المستخدم يدخل رقم بناءً على الرقم ف الاسكريبت هينفذ أوامر مختلفة زي pwd أو cd وpwd، أو يظهر رسالة "Not Understand" لو الرقم مش صحيح.

```
script.sh                                                    bash
#!/bin/bash
read  -p "Enter you Command Number : " var
case $var in
1 )
pwd
;;
2)
cd
pwd
;;
*)
echo "Not Understand"
;;
esac
```

- شرح الاسكريبت اول سطر بيطلب من ال user انه يدخل رقم والرقم اللي ال user هيدخله هيتخزن في Variable ف لو الرقم اللي دخله ال user كان 1، الاسكريبت هيقوم بتنفيذ الأمر pwd. لو الرقم اللي دخله ال user كان 2، الاسكريبت هيقوم بتنفيذ الأمر cd و pwd. ولو الرقم اللي دخله ال user مش 1 ولا 2، الاسكريبت هينفذ الأوامر الموجودة تحت * وهي Not Understand

```
mohamed@MohamedAtef:~$ ./script.sh
Enter you Command Number : 1
/home/mohamed/Desktop/shell/Documents/d
```

➢ مثال تاني

```
script.sh                                                    bash
#!/bin/bash

case $1 in
"mohamed" )
      echo "Big Boss"
;;
"Ali" )
      echo "mini Boss"
;;
*)
     echo "Normal User"
;;
esac
```

➢ عند تنفيذ الاسكريبت

```
mohamed@MohamedAtef:~$ ./script.sh  mohamed
Big Boss
```

Regular Expressions

- ال **Regular Expressions** او ال Pattern Matching او مايعرف ب Regex نقدر نقول انها عباره عن Matching Criteria او طريقه احنا بنسخدمها علشان ن Match بيها Using Pattern معين علي اي file موجود عندنا . ودي بعض Regular Expressions اللي بنستخدمها

Symbol	Descriptions
.	replaces any character
^	matches start of string
$	matches end of string
*	matches up zero or more times the preceding character
\	Represent special characters
()	Groups regular expressions
?	Matches up exactly one character
!	Match all strings except any of the patterns
@	Match exactly one occurrence of any of the patterns
+	Match one or more occurrence of any of the patterns

- هنطبق علي ال **Regular Expressions** بأستخدام ال CLI ونشوف ازاي هنستخدمها مع ال Case Command

➢ ممكن مثلا اقوله اعملي List علي اي file بيبدء بحرف f ف هستخدم ال asterisk *

```
mohamed@MohamedAtef:~$ ls f*
fa    fbc    fcde    file    file1    file2    file22   felib
```

➢ لو انا عايز اعمل list ل اي File بيتكون من 3 حروف وانا مش عارف ال 3 حروف دول ف هستخدم question mark ?

```
mohamed@MohamedAtef:~$ ls ???
abc    fbc    bcd    fab
```

➢ او لو انا عارف مثلا اول 4 حروف ومش عارف الحرف الخامس هستخدم question mark ?

```
mohamed@MohamedAtef:~$ ls file?
file1    file2    file3    file4    filem
```

➢ لو انا عايز اعمل List ل اي file بيبدء بحرف ال a او حرف ال b هستخدم Brackets [ ]

```
mohamed@MohamedAtef:~$ ls [ab]*
abc    bcd
```

- لو انا عايز اعمل العكس ان اعمل List ل اي file مبيبدئوش بحرف ال a او حرف ال b هزود علامه !

```
mohamed@MohamedAtef:~$ ls [!ab]*
cdb   cca  fa    fbc    fcde file     file1    file2    file22   felib
```

- لو انا عايز اديله range بمعني ان انا عايزاعمل List لكل ال files اللي بتبدء بحرف ال a وال b وال c

```
mohamed@MohamedAtef:~$ ls [a-c]*
abc    bcd    cda
```

- لو انا عايز اعمل العكس ان انا هديله range ويعملي List ماعدا الموجوده في ال range ده هستخدم ^

```
mohamed@MohamedAtef:~$ ls [^a-c]*
def     vd     file1     file2
```

- لو انا عايز اعمل List ل اي file بيبدء مثلا بكلمه file وينتهي بحرف a او b هستخدم [ab]

```
mohamed@MohamedAtef:~$ ls file[ab]
filea          fileb
```

- لو انا عايز اعمل Search علي اي file موجود عندي بيبدء بأي حرف وبينتهي بأي حرف وبيحتوي علي space

```
mohamed@MohamedAtef:~$ ls *[[:space:]]*
'shell script'
```

- لو انا عايز اعمل Search علي اي file بيتكون من خمس حروف اول 4 حروف اسمهم file والحرف الخامس بيكون alpha numeric سواء كان حرف او رقم

```
mohamed@MohamedAtef:~$ ls file[[:alnum:]]
file1          file2    filea    filec    filef    file3
```

- لو هستخدم ال **Regular Expressions** مع ال case command في الاسكريبت لازم اخد بالي من حاجه ان ال Bash مبيدعمش بعض ال sub pattern زي مثلا (Pattern)? و (Pattern)* و (Pattren)@ و (Pattern)+ و (Pattern)! ف علشان اقدر استخدم ال sub Pattern دي لازم اضيف Two line في الاسكريبن وهما **shopt -s extglog** و **export LC_COLLATE=C**

- ف ال **shopt -s extglob:** ده أمر بتكتبه عشان تفعّل ميزة في Bash بتخليك تقدر تستخدم pattern matching زي +(pattern) أو !(pattern). الحاجات دي مش شغالة بشكل افتراضي، بس لما تشغلها تقدر تطابق حاجات أكتر بسهولة في الاسكربت بتاعك.

- **export LC_COLLATE=C:** ده أمر بتستخدمه عشان تخلي ترتيب الأحرف في المطابقة يعني بتخليه يفرق مابين ال a Small وال A Capital (زي لما تستخدم [a-z] أو [A-Z]) يبقى بناءً على ترتيب ASCII، يعني الترتيب اللي هو مش مرتبط باللغة. ده بيخليك متأكد إن النطاقات دي شغالة زي ما أنت متوقع من غير ما تأثر عليها اللغة اللي شغال عليها الجهاز.

```
shopt -s extglob

export LC_COLLATE=C
```

- في المثال انا عايز ال User يدخلي Input وبناءا عليه هيقول لل user ان ال Input اللي هو مدخلهوله اذا كان lower case ولا upper case ولا Integer

```bash
script.sh
#!/bin/bash
shopt -s extglob
export LC_COLLATE=C
case $1 in
@([a-z]) )
        echo "lower Case"
;;
@([A-Z]) )
       echo "Upper Case"
;;
@([0-9]) )
       echo "Integer"
;;
esac
```

- انا هنا مدخله حرف A وهو هيقولي اذا كان lower او upper او Integer

```
mohamed@MohamedAtef:~$ ./script.sh  A
Upper Case
```

Looping Commands

- فكره ال Loops زي اي Programming Language ان انت لو عندك Block of code او مجموعه من ال Commands عايز تعملها repeat او تعملها Execute multiple Times بنستخدم معاها ال Loops. فعندنا في ال Bash Scripting اكتر من طريقه ن Create بيها Loop جوه الاسكريبت بتاعنا

1- اول طريقه انك تستخدم ال For Loops

- ودي بتستخدم عندما تريد تنفيذ أمر أو مجموعة من الأوامر لعدد محدد من المرات أو عند التنقل عبر قائمة من القيم. عن طريق ان انت بتستخدمها بال Syntax ده ان انت تقوله **For** وبعدين ال **Argument** بتاعك وبعدين **in** وبعدين بتديله **Sequence** معين سواء كان ال Sequences ده عباره عن List او مجموعه من ال Numbers وبعدين تقوله **do** وتديله ال **Commands** اللي انت عايز تنفذها وفي الاخر تقوله **done** ف اللي هو هيعمله هيعمل iterates وهيعمل execute لل Commands دي Multiple time according لل Sequence اللي انت حططهوله

```
for arg in [list]
do
command1
command2
command3
done
```

- ممكن ان انت تسخدم ال Loops من خلال ال CLI او من خلال ان احنا نكتب الاسكريبت في ملف
- في المثال ده احنا عرفنا Variable اسمه NUMBERS وحطينا جواه مابين ال Double Quotes الارقام 1 2 3 4 5 ف لما احنا قولناله for وبعدين قولناله ان ال Argument هيكون number وبعدين In وبعدين $NUMBERS اول اما اضغط Enter هيفتحلي Prompt > بعد كده علشان انفذ ال commands اللي انا عايزها قولتله do وبعدين قولتله ان انا عايز انفذ echo -n "$number" ان هو هيعرضلي الارقام الموجوده في $number

```
mohamed@MohamedAtef:~$ NUMBERS="1 2 3 4 5"
mohamed@MohamedAtef:~$ for number in $NUMBERS
> do
> echo -n "$number"
> done
12345
```

- **مثال تاني** . في المثال ده هنعمل create ل مجموعه من ال Files باستخدام ال for loop وهنعمل كمان list لل Files

```
mohamed@MohamedAtef:~$ for i in 1 2 3 4 5
> do
> touch file_$i
> ls
> done
file_1
file_1  file_2
file_1  file_2  file_3
file_1  file_2  file_3  file_4
file_1  file_2  file_3  file_4  file_5
```

- ممكن ان احنا نطبق اللي عملناه من خلال ان احنا هنعمل اسكريبت ان احنا هنعمل file وليكن test.sh وهنكتب جواه الاسكريبت اللي عايزين ننفذه

```
script.sh                                                    bash
#!/bin/bash

for i in 1 2 3 4 5
do
   touch file_$i
   ls
done
```

- لو نفذنا ال Script هنلاقيه عمل Create لل Files

```
mohamed@MohamedAtef:~$./ script.sh
file_1
file_1  file_2
file_1  file_2  file_3
file_1  file_2  file_3  file_4
file_1  file_2  file_3  file_4  file_5
```

2-تاني طريقه اني استخدم ال While Loop

- ال While عباره عن Conditional Loop زي ال For Loop بس في ال While بيكون موجود Condition ف as long as ان ال Condition ده ب True ال While Loop بتاعتك هتفضل ت Iterate لحد اما ال Condition بتاعك يبقي ب False ساعتها هتعمل break لل Loop بتاعتك
- ال Syntax الخاص بال While Loop هو شبه ال For Loop بس بدل اما بتديله ال Sequence بتديله Condition معين

```
while [  condition  ]
do
   command1
   command2
   command3
done
```

- مثال . في المثال ده انا عملت Initialize ل Variable اسمه counter وحطيت جواه 1 وقولتله while ان ال counter اقل من 10 روح اعمل iterate واعمل Run لل Code اللي هو اعمل echo لل counter واعمل increment لل 1

```
script.sh                                                    bash
#!/bin/bash

counter=1
while [ $counter -le 5 ]
do
   echo $counter
   let counter+=1
done
```

- عند تنفيذ الاسكريبت هيكون ال Output

```
mohamed@MohamedAtef:~$ ./script.sh
1
2
3
4
5
```

3- تالت طريقه اني استخدم ال Until Loop

- ال Until عكس ال While ان ال Until بتفضل ت Iterate طالما ال Condition بتاعك ب False

```
script.sh                                                    bash
#!/bin/bash

counter=1
untile [ $counter -gt 5 ]
do
   echo $counter
   let counter+=1
done
```

- عند تنفيذ الاسكريبت هيكون ال Output

```
mohamed@MohamedAtef:~$ ./script.sh
1
2
3
4
5
```

Exit and exit status

- ال Exit هو عباره عن Command احنا بنستخدم علشان نعمل Terminate للاسكريبت . بمجرد ان انت بتعمل Terminate ل عمل الاسكريبت بيكون فيه قيمه اسمها Exit Value او Status Value بتوضح كيفيه انتهاء هذا الاسكريبت هل اتعمل ليه Terminate بشكل صحيح او لا
- وبيكون عندي قيمتين في ال Exit Value
  - لو الاسكريبت اتعمل ليه Terminate بشكل صحيح هيكون ال Exit Value ب Zero(0) ودي معناها Successful
  - لو الاسكريبت اتعمل ليه Terminate بشكل غير صحيح وحصل اي مشكله هيكون ال Exit Value ب Non Zero ودي معناها Unsuccessful او Failure
- لو انا عايز اعرف ال Exit Value بضيف echo $? في نهايه الاسكريبت او ان انا لما انفذ الاسكريبت او اي Command هكتب echo $? في ال CLI بعد تنفيذ ال command وهو هيعرضلك قيمه ال Exit Value اذا كانت ب zero او Non Zero

```
echo $?
```

➢ **مثال** لو انا نفذت ال command ده "Hello, world!" echo وعايز اعرف ال Exit Value هستخدم echo $? هنلاقيه قايلنا ان ال Exit Value ب 0 معناها ان ال command اتعمله execute حالا بيكون Successful

```
mohamed@MohamedAtef:~$ echo "Hello, World!"
Hello, World!
mohamed@MohamedAtef:~$ echo $?
0
```

➢ لو انا نفذت اي command مش موجود زي مثلا Devops هيقولك Not Found ولو شوفنا ال Exit Value باستخدام echo $? هنلاقيه قايلنا ان ال Exit Value ب 127 معناها ان ال command اللي انت عملته execute حالا بيكون Unsuccessful

```
mohamed@MohamedAtef:~$ Devops
Devops: command not found
mohamed@MohamedAtef:~$ echo $?
127
```

- لو انت بت Execute Script معين الاسكريبت ده بي return exit value لل Terminal اللي عندي بعد اما يخلص . والاسكريبت بي Return ال Exit Status بتاعت اخر Command انت عملتله Execute في الاسكريبت

➢ مثال . لو انت عملت execute للاسكربت ده الاسكريبت ده عباره عن انه هيعمل create ل file1 و dir1 وبعدين هيعمل list لل files وال Directories الموجوده ف الاسكريبت ه return ال Exit status بتاعت اخر Command في الاسكريبت وهو في حالتنا هنا ال ls

```
script.sh                                                    bash
#!/bin/bash

touch file1
mkdir dir1
ls
```

➢ ف لو عملنا execute هنلاقيه عمل create ل dire1 و file1 وبعدين عمل list ولو شوفنا ال Exit Value باستخدام echo $? هنلاقي رجعلنا ال Exit Status اللي هي 0 بتاعت اخر command انت عملتله Execute اللي هو ال ls

```
mohamed@MohamedAtef:~$ ./ script.sh
dir1  file1  script.sh
mohamed@MohamedAtef:~$ echo $?
0
```

- ال Exit زي ماوضحنا قبل كده هو عباره عن Command احنا بنستخدمه علشان نعمل Terminate للاسكريبت بمعني ان انا لو كتبت exit في نهايه الاسكريبت هو هينفذ الاسكريبت وبعدين اول اما يلاقي exit هيعمل Terminate للاسكريبت وساعتها هي return لل CLI ال Execute status بتاعت اخر Command انت عملتله Execute قبل ال Exit
- او ممكن ان تكتب في نهايه الاسكريبت exit $? ودي equivalent لل exit
- او ممكن ان انت تكتب exit وبعدين تديله اي Integer Value زي مثلا exit 10 ساعتها هو هيرجع ال Integer Value دي لل Terminal

```
#!/bin/bash
COMMAND _1
. . .
#will exit with status of last command.
COMMAND_LAST

exit
```

```
#!/bin/bash
COMMAND _1
. . .
#will exit with status of last command.
COMMAND_LAST

exit $?
```

The Select Command and Menus

- ال select loop بتعمل Create ل Menus ان بيتحلك ان انت تعرض قائمة من الخيارات على المستخدم، والمستخدم يختار منها بالرقم. وبيكون ال Syntax بتاعه بالشكل ده

```
select variable in list
do
  commands
done
```

- variable: ده المتغير اللي هيخزن القيمة اللي المستخدم هيختارها.
- list: القائمة من الخيارات اللي هتعرضها على المستخدم.
- commands: الأوامر اللي هتتنفذ بناءً على اختيار المستخدم.

➢ مثال علي ال Select

```
script.sh                                                    bash
#!/bin/bash

select name in mohamed ahmed Mahmoud
do
   case $name in
        "mohamed" )
                echo "Mohamed is good boy"
                break
         ;;
        "ahmed" )
                echo "Ahmed is the best"
                 break
         ;;
        "Mahmoud" )
                 echo "Mahmoud is a bad boy"
                 break
         ;;
         *)
                echo "Integer"
                break
    esac
done
```

➢ عند تنفيذ الاسكريبت هيعرضلي ال List اللي انا مختارها ومن خلالها اقدر اختار منها ف لو انت اختر 1 اللي هي بتعبر عن Mohamed هيظهرلك Mohamed is a good boy وهكذا للباقي

```
mohamed@MohamedAtef:~$ ./script.sh
1) mohamed
2) ahmed
3) mahmoud
#? 1
Mohamed is a good boy
```

- مثال تاني علي ال Select بشكل اوسع .الاسكريبت ده بيعمل List بخيارات لل user علشان يختار منهم . لو انت هتعمل create Folder هيسألك عن اسم ال Folder، ولو مش موجود بيعمله، ولو موجود بيقول لك إنه موجود بالفعل.ولو انت هتعمل create File هيسألك عن اسم ال file، ولو مش موجود بيعمله، ولو موجود بيقول لك إنه موجود بالفعل. وعندك كمان من ضمن ال list لو انت عايز تعرض الملفات تعمل displayFiles وده بيعرض كل الملفات الموجودة في الcurrent directory . ولو انت عايز تعرض ال directory ان اتعمل displayFolders وده بيعرض كل المجلدات الموجودة في ال current directory..

script.sh bash

```bash
#!/bin/bash

select name in createFolder createFile displayFiles displayFolders
do
   case $name in
      "createFolder" )
         read -p "Enter Name of dir: " name_dir
         if [[ -d $name_dir ]]; then
            echo "Directory already exists"
         else
            mkdir $name_dir
            echo "Folder created"
         fi
         break
         ;;
      "createFile" )
         read -p "Enter Name of file: " name_file
         if [[ -f $name_file ]]; then
            echo "File already exists"
         else
            touch $name_file
            echo "File created"
         fi
         break
         ;;
      "displayFiles" )
         find . -type f
         break
         ;;
      "displayFolders" )
         ls -F | grep "/"
         break
         ;;
      *)
         echo "Invalid option"
         break
         ;;
   esac
done
```

- لما تشغل الاسكربت، هيظهرلك list بالخيارات الأربعة.تختار الرقم المرتبط بالعملية اللي عايز تنفذها، وبناءً على اختيارك، الاسكربت هيطلب منك تدخل اسم الملف أو المجلد لو كنت بتختار createFolder أو createFile. في حالة اختيار displayFiles أو displayFolders، الاسكربت هيعرضلك الملفات أو المجلدات الموجودة ف.ولو اخترت خيار مش موجود، الاسكربت هيقول لك "Invalid option" وهيخرج.

```
mohamed@MohamedAtef:~$ ./script.sh
1) createFolder
2) createFile
3) displayFiles
4) displayFolders
#? 4
dir1/
```

Debugging Your Scripts

- ال **Debugging** احنا بنستخدمها علشان نكتشف ونصحح ال errors اللي ممكن تحصل في الاسكريبت والهدف الاساسي انه بيتاكد ان ال Script بيشتغل بشكل صحيح بدون اي مشاكل
- ف احنا عندنا اكتر من طريقه علشان اعمل Debugging

1- اول طريقه ان احنا واحنا بنفذ الاسكريبت نستخدم **bash -x** قبل الاسكريبت زي مثلا **bash -x script.sh** ده كده هينفذلك الاسكريبت وهيعرضلك كل الخطوات اللي تمت في الاسكريبت ولو فيه اي error هيعرضلك المشكله في انهي line

- في المثال التالي موجود في الاسكريبت مجموعه من الاوامر وعايزين نعمل Debugging ونتأكد هل الاسكريبت فيه اي مشكله او لا ف هنستخدم **bash -x script.xh**

script.sh bash

```
#!/bin/bash

y=5
echo "$y"
echo "Hello Mohamed"
echo "Test"
```

- لما عملنا Debugging للاسكريبت ظهرلي ال output بالشكل ده ان هو عرضلك كل الاوامر بخطوات التنفيذ ف لو فيه اي مشكله انت هتقدر تعرف المشكله دي فين وبسبب ايه

```
mohamed@MohamedAtef:~$ bash -x  ./script.sh
+ y=5
+ echo 5
5
+ echo 'Hello Mohamed'
Hello Mohamed
+ echo Test
Test
```

- ولو احنا عايزين نعمل Debugging بدون تشغيل الاسكربت ان لو فيه مشكله هيعرضلك المشكله علي طول هستخدم **bash -n** وبعدين اسم الاسكريبت زي مثلا **bash -n script.sh**

2- تاني طريقه ان انت ممكن تستخدم **set -x** في بدايه الاسكريبت علشان تعمل Start لل Debugging ولو انت عايز تعمل End لل Debugging هتستخدم **set +x** ولو انت عايز توقف الاسكريبت عند اول خطأ يظهر هتستخدم **set -e**

- هنستخدم نفس الاسكريبت وهنستخدم ال set -x وال set +x

script.sh bash

```
#!/bin/bash
set -x
y=5
echo "$y"
echo "Hello Mohamed"
set +x
echo "Test"
```

- لما عملنا Debugging للاسكريبت ظهرلنا ال output بالشكل ده ان هو عرضلك كل الاوامر بخطوات التنفيذ ولكن علشان استخدمنا set +x ان احنا عملنا stop لل Debugging ف هو وقف عمل ال Debugging علي ال echo “Test” وطبعلك Test علي طول بدون عمل Debugging عليه

```
mohamed@MohamedAtef:~$./script.sh
+ y=5
+ echo 5
5
+ echo 'Hello Mohamed'
Hello Mohamed
+ set +x
Test
```

Arrays

- ال Arrays في ال Shell زيها زي أي array في لغات برمجة تانية، بتخليك تخزن كذا قيمة في متغير واحد. يعني بدل ما تعمل متغير لكل قيمة، ممكن تحط القيم كلها في array وتتعامل معاها بسهولة.
- **ومن خصائص ال Arrays في ال Shell Script**

- ان ال Array بتكون One-dimensional array يعني كل القيم بتتحط في Line واحد، وال Array دي ممكن يكون فيها لحد 1024 عنصر
- و ال Index بيبدأ ب Zero
- ممكن تضيف أو تحذف عنصر (set/unset element): كل عنصر في ال Arrays تقدر تضيفه أو تمسحه بشكل فردي
- مفيش ترتيب معين لازم تمشي عليه (Values don't need to be in order): يعني ممكن تضيف العناصر بأي ترتيب

- علشان اعمل Set ل Value معينه ل element في Array عن طريق ان انا هستخدم

```
mohamed@MohamedAtef:~$ array[0]=mohamed
mohamed@MohamedAtef:~$ array[1]=atef
mohamed@MohamedAtef:~$ array[2]=ahmed
```

- علشان اعمل Print لل Value بتاعت ال array elements هستخدم echo وبعدين dollar sign $ وبين { } Curly Braces هكتب اسم ال Array ومابين [ ] Square Brackets بكتب ال Index

```
mohamed@MohamedAtef:~$ echo ${array[0]}
mohamed
mohamed@MohamedAtef:~$ echo ${array[1]}
atef
mohamed@MohamedAtef:~$ echo ${array[2]}
ahmed
```

- وممكن ان انا اكتب كل ال Value في Array واحد زي مثلا

```
mohamed@MohamedAtef:~$ myarray=(1 2 3 4 5 "Mohamed" "Atef")
```

- لو انا عملت print لل Array بدون تحديد اي Index هيطبعلي اول Index وهو 1

```
mohamed@MohamedAtef:~$ echo $myarray
1
```

- لو انا عايز اطبع كل ال Array هستخدم @ او ال * علامه ال @ و * بتستخدم لو انا عايز اطبع كل ال elements الموجوده في ال Array

```
mohamed@MohamedAtef:~$ echo ${myarray[@]}
1 2 3 4 5 Mohamed Atef
```

- لو انا عايز اعرف ال Index بتاعت ال Array بادئه من كام ل كام هكتب قبل اسم ال array هكتب علامه التعجب !

```
mohamed@MohamedAtef:~$ echo ${!myarray[@]}
0 1 2 3 4 5 6
```

- لو انا عايز اعرف Length ال Arrays هستخدم علامه الهاش #

```
mohamed@MohamedAtef:~$ echo ${#myarray[@]}
7
```

- لو انا عايز اعمل Slicing لل Array يعني مثلا في اول مثال انا عايز اعمل تقطيع لل Array بدايه من ال Index رقم 2 وهاتلي عنصرين من ال Array بدايه من ال Index رقم 2

```
mohamed@MohamedAtef:~$ echo ${myarray[@]:2:2}
3 4
mohamed@MohamedAtef:~$ echo ${myarray[@]:5:2}
Mohamed Atef
mohamed@MohamedAtef:~$ echo ${myarray[@]:5:3}
Mohamed Atef
```

Change and Update Array

1- Append

- علشان اضيف اي element في ال Array هكتب اسم ال Array وبعدين + وبعدين = equal وبين قوسين هضيف كل ال elements اللي انا عايزها انا هنا ضفت Ali و Abdo

```
mohamed@MohamedAtef:~$  myarray+=("Ali" "Abdo")
7
```

- لو عرضنا ال Array تاني هنلاقيه ضاف ال elements اللي احنا ضفناها

```
mohamed@MohamedAtef:~$  echo ${myarray[@]}
1 2 3 4 5 Mohamed Atef Ali Abdo
```

2- Delete

- علشان امسح element من ال Array هستخدم unset وبعدين اسم ال Array وبين Square Brackets [ ] هكتب رقم ال Index الخاص بال element

```
mohamed@MohamedAtef:~$  unset myarray[0]
```

Functions

- علشان اعرف Function في ال BASH فيه عندي طريقتين

1- اول طريقه ان انا بديله اسم ال Function وبعدين قوسين ()وبعدين بين { } Curly Braces بحط كل ال Commands اللي انا عايز انفذها وعلشان ال function تشتغل لازم اضيف في نهايه ال Function ال Call ودي عباره عن انا بكتب اسم ال Function بدون قوسين

```
function_name() {

 # Commands that will be executed when the function is called
}
# Call the function
function_name
```

➢ مثال

```
script.sh                                                    bash
#!/bin/bash
hello(){
    echo "Hello Mohamed Atef"
}
hello
```

➢ عند تنفيذ ال Function

```
mohamed@MohamedAtef:~$ ./ script.sh
Hello Mohamed Atef
```

2- تاني طريقه ان انا بكتب function وبعدين بين اسم ال function بدون قوسين() وبعدين بين Curly Braces { } بحط كل ال Commands اللي انا عايز انفذها وعلشان ال function تشتغل لازم اضيف في نهايه ال Function ال Call ودي عباره عن انا بكتب اسم ال Function بدون قوسين

```
function_name() {

# Commands that will be executed when the function is called
}
# Call the function
function_name
```

➢ مثال

```
script.sh                                                    bash
#!/bin/bash
function hello{
    echo "Hello Mohamed Atef"
}
hello
```

Variables Scope in Functions

- فيه عندي نوعين من ال Variables Scope اول نوع وهو ال Global Scope وتاني نوع وهو ال Local Scope

Global Scope -1

- اي Variable بيكون بره ال Function بيكون Global variables ودي بنقدر نستخدمها في اي مكان في الاسكريبت سواء كان جوه ال Function او بره ال Function

```bash
#!/bin/bash
# Global variable
my_var="Global Variable"

function print_var {
    echo "Inside function: $my_var"
}
print_var   # Will print: Inside function: Local Variable
```

Local Scope -2

- وهنا اي Variable بتتعرف جوه ال Function بتكون Local Variable ودايما بنستخدم كلمه local وال Variable دي محدش يقدر يشوفها او يستخدمها الا جوه ال Function اللي اتعرفت فيها وال Variable بتكون موجوده طول ما الداله شغاله واول اما ال Function يخلص ال Variable ده بيختفي

```bash
#!/bin/bash

my_var="Global Variable"

function print_var {
    local my_var="Local Variable"
    echo "Inside function: $my_var"
}

print_var   # Will print: Inside function: Local Variable
echo "Outside function: $my_var"   # Will print: Outside function: Global Variable
```

➢ مثال علي الطريقتين

```bash
#!/bin/bash
x=50
y=30

test_scope() {
    x=20
    z=10
    local y=10
    echo "Inside Function x= $x , y= $y , z= $z"
}

echo "Before Call Function x= $x , y= $y , z= $z"
test_scope
echo "After Call Function x= $x , y= $y , z= $z"
```

➢ لما ننفذ الاسكريبت

```
mohamed@MohamedAtef:~$  ./ script.sh
Before Call Function x= 50 , y= 30 , z=
Inside Function x= 20 , y= 10 , z= 10
After Call Function x= 20 , y= 30 , z= 10
```

What is sed?

- ال Sed عباره عن Stream Editor بمعني انه Non-Interactive Editor ان انت تقدر تعدل في ال File بتاعك من غير متحتاج تفتح ال file وبالتالي تقدر تعمل Automation لل Task اللي انت عايز تعملها
- اي حاجه انت بتعملها بال vi بتقدر تعملها بال sed
- By default, ال sed مبيغيرش ال file الاصلي هو بيعمل تغيير وبيعرضهالك في ال Display
- لو انت عايز ال sed يغير في ال file الاصلي هتستخدم اوبشن معين وهنشوف الاوبشن ده في الشرح

Syntax:

```
sed 'command' filename
```

- ال Command بيتقسم ل حاجتين ال Action اللي انت عايز تعمله وال Address بتاع ال Line اللي انت هتنفذ عليه ال Task بتاعك
- ال Action ممكن يكون
  - substituteاللي هو ال Search وال Replace
  - او ممكن يكون delete انك تعمل delete لسطر معين
  - او ممكن يعمل Print لسطر معين
  - او ممكن يغير في السطر
- ال Address بيكون عنوان السطر اللي انت عايز تغير فيه
  - ف ممكن يكون Address صريح ف تديله Line Number او ممكن تديله Range
  - او ممكن يكو ن ال Address بي Match علي Regular Expressions ان هو بي Match علي السطور اللي فيها Pattern معين
  - او ممكن يكون الاتنين مع بعض ال Line Number وال Regular Expressions

- المثال اللي احنا هنطبق عليه هيكون عباره عن file اسمه geekfile.txt وبيحتوي علي

```
unix is great os. unix is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

1- Replacing or substituting string

```
sed 's/search pattern/replace pattern/' filename
```

- لو انا عايز اعمل replace ل Pattern معين هستخدم sed ومابين single quotes اكتب حرف ال s وده معناه ان انا هعمل substitute وبعدين مابين ال / slash هكتب اسم ال Pattern اللي انا عايز اعمله replace وبعدين / slash وبعدين هكتب ال Pattern اللي انا عايز اضيفه وبعدين اسم ال file

➢ في المثال التالي انا عايز اعمل replace لل Pattern اللي اسمه unix ل windows ف كل اللي هيعمله هيعمل search في ال file اللي اسمه geekfile.txt عن اي line موجود فيه كلمه unix ويعملها replace ل windows بس اللي هتلاحظه انه عمل replace بس ل اول Pattern موجود في ال Line لان ده ال default لل sed انه بيعمل match ل اول pattern موجود في ال Line

```
mohamed@MohamedAtef:~$ sed  's/unix/windows/'  geekfile.txt
windows is great os. unix is opensource. unix is free os.
learn operating system.
windows linux which one you choose.
windows is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

2- Replacing all the occurrence of the pattern in a line

➢ طب لو انا عايز اعمل replace لكل ال occurrences الموجوده في ال line هستخدم اوبشن /g وده معناه Global

```
mohamed@MohamedAtef:~$ sed  's/unix/windows/g'  geekfile.txt
windows is great os. windows is opensource. windows is free os.
learn operating system.
windows linux which one you choose.
windows is easy to learn.windows is a multiuser os.Learn windows .windows is a powerful.
```

3- Replacing the nth occurrence of a pattern in a line

- لو انت مثلا عندك line وال Line ده موجود فيه اكتر من Pattern وليكن ال pattern فيه اكتر من كلمه unix وعايز تعمل مثلا replace ل تاني كلمه unix فقط هتستخدم 2/ ف هو هيعمل search في ال line ويشوف تاني pattern ويعمله replace وهكذا

```
mohamed@MohamedAtef:~$ sed 's/unix/windows/2' geekfile.txt
unix is great os. windows is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.windows is a multiuser os.Learn unix .unix is a powerful.
```

4- Replacing from nth occurrence to all occurrences in a line

- لو اخدت بالك في المثال السابق هتلاقيه عمل replace ل تاني pattern بس ف لو انت عايز تعمل replace من اول تاني Pattern مثلا واي Pattern بعديه هتستخدم 2g/

```
mohamed@MohamedAtef:~$ sed 's/unix/windows/2' geekfile.txt
unix is great os. windows is opensource. windows is free os.
learn operating system.
unix linux which one you choose.
unix is easy to learn.windows is a multiuser os.Learn windows .windows is a powerful.
```

5- Replacing string on a specific line number

- لو عايز اعمل replace في line معين وليكن عايز اعمل replace لكلمه unix في السطر رقم 3

```
mohamed@MohamedAtef:~$ sed '3 s/unix/windows/' geekfile.txt
unix is great os. unix is opensource. unix is free os.
learn operating system.
windows linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

6- Replacing string on a range of lines

- لو انا عايز اعمل replace في range معين وليكن عايزه يعمل replace من السطر رقم 1 ل السطر رقم 3 هستخدم 1,3

```
mohamed@MohamedAtef:~$ sed  '1,3 s/unix/windows/'  geekfile.txt
windows is great os. unix is opensource. unix is free os.
learn operating system.
windows linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

- لو انا عايزه يعمل replace من السطر الاول ل السطر الاخير هستخدم 1,$

```
mohamed@MohamedAtef:~$ sed  '1,$ s/unix/windows/'  geekfile.txt
windows is great os. unix is opensource. unix is free os.
learn operating system.
windows linux which one you choose.
windows is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

7- Duplicating the replaced line with /p flag

- لو انت عايزه يكرر ال Line اللي عمل ليه replace هتستخدم /p

```
mohamed@MohamedAtef:~$ sed  '3 s/unix/windows/'  geekfile.txt
unix is great os. unix is opensource. unix is free os.
learn operating system.
windows linux which one you choose.
windows linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

8- Printing only the replaced lines

- لو انت عايزه يعمل display لل Lines اللي عمل فيها replace بس هتستخدم اوبشن -n علشان يلغي الطباعه التلقائيه يعني انت اللي هتتحكم في اللي هيتعمل ليه display وبعدين هتستخدم /p علشان تطبع ال lines اللي حصل فيها تغيير بس لو انت استخدمت اوبشن -n بس ومستخدمتش /p مش هطبيع اي حاجه

```
mohamed@MohamedAtef:~$ sed  -n  '3 s/unix/windows/p'  geekfile.txt
windows linux which one you choose.
```

9- Deleting lines from a particular file

- لو انت عايز تعمل delete لسطرمعين وليكن عايز تعمل delete للسطر رقم 2 هتستخدم 2d

```
mohamed@MohamedAtef:~$ sed '2d' geekfile.txt
unix is great os. unix is opensource. unix is free os.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

- لو انت عايز تعمل delete ل اخر سطر هتستخدم $d

```
mohamed@MohamedAtef:~$ sed '$d' geekfile.txt
unix is great os. unix is opensource. unix is free os.
learn operating system.
unix linux which one you choose.
```

- لو انت عايز تعمل delete ل range معين هتستخدم

```
mohamed@MohamedAtef:~$ sed '1,3d' geekfile.txt
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

- لو انت عايز تعمل delete من سطر معين ل اخر سطر

```
mohamed@MohamedAtef:~$ sed '1,$d' geekfile.txt
```

- لو انت عايز تعمل delete للسطر اللي فيه Pattern معين وليكن عايز تعمل delete للسطر اللي فيه كلمه unix

```
mohamed@MohamedAtef:~$ sed '/unix/d' geekfile.txt
learn operating system.
```

10- Edit the file in place without printing to the console (overwrite the file)

- لو انت عايز اي تغيير تعمله باستخدام ال sed يطبق علي ال file الاصلي هتستخدم اوبشن -i

```
mohamed@MohamedAtef:~$ sed -i '/unix/d' geekfile.txt
```

11- Multiple commands syntax

- لو انت عايز تنفذ اكتر من command في نفس السطر هتستخدم قبل كل command اوبشن e-

```
mohamed@MohamedAtef:~$ sed -e '/unix/d' -e '/linux/d' -e '1,$ s/unix/windows/' geekfile.txt
```

- لو انت عندك اكتر من command مثلا ومش عايز كل شويه تكتب e- ممكن ان انت تكتب كل ال command في file ويكون كل command في سطر وعلشان تنفذ ال commands الموجوده في ال file هتستخدم اوبشن f-

```
mohamed@MohamedAtef:~$ sed -f actionfile geekfile.txt
```

- لو انت عايز تعمل display لكل ال Lines اللي بيبدئوا بكلمه معينه او حروف معينه هتضيف ^ قبل الكلمه

```
mohamed@MohamedAtef:~$ sed -n '/^uni/p' geekfile.txt
unix is great os. unix is opensource. unix is free os.
unix linux which one you choose.
unix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.
```

- لو انت عايز تعمل display لكل ال Lines اللي بينتهو بكلمه معينه او حروف معينه هتضيف $ بعد الكلمه

```
mohamed@MohamedAtef:~$ sed -n 'choose$/p' geekfile.txt
unix linux which one you choose.
```

- ممكن كمان تقوله انا عايز اعمل display لل line اللي بيبدء مثلا بكلمه unix وبينتهي بكلمه powerful

```
mohamed@MohamedAtef:~$ sed -n 'choose$/p' geekfile.txt
unix linux which one you choose.
```

What is AWK?

- ال AWK عباره عن Programming Language الغرض منها ان احنا نستخدم المعلومات الموجوده في ال files وال Configuration files اللي عندنا في السيستم والغرض الاساسي منها ان احنا ن generate reports
- ف ال AWK بيشتغل عن طريق انه بيعدي في ال file علي سطر سطر وبت Search فيهم ان مين فيهم اللي بي Match علي Pattern معين او Regular Expression معين وبعد كده بتنفذ عليه Action معين

- **WHAT CAN WE DO WITH AWK ?**
  1. AWK Operations :
     - Scans a file line by line
     - Splits each input line into fields
     - Compares input line/fields to pattern
     - Performs action(s) on matched lines
  2. Useful For :
     - Transform data files
     - Produce formatted reports
  3. Programming Constructs :
     - Format output lines
     - Arithmetic and string operations
     - Conditionals and loops
     -

- **What is the main purpose of awk?**
  - Process text files: Analyzing and manipulating data in text files.
  - Pattern scanning and processing: Searching for patterns within files and performing specified actions when those patterns are found.
  - Data extraction and reporting: Generating formatted reports from a text database or any structured data.
  - Text transformation and manipulation: Efficiently handling and transforming data or text within files.

- **Syntax :**

```
awk 'BEGIN { statements } { statements } END { end statements }'
```

- ال { statements } BEGIN ده اوبشن ممكن تكتبه وممكن متكتبهوش
- ال { end statements } END ده اوبشن ممكن تستخدمه او متستخدمهوش
- اللي هيتنفذ اللي مابين ال Curl Practice { }
- ال BEGIN فايدته انه بيتنفذ قبل اما يقرأ اي سطر علشان كده ال BEGIN بنستخدمه لو انا عايز اعمل Variable Initialization او اعرف Variable واديله قيمه او ارسم header لل report
- ال END بيتنفذ بعد اما يقرأ كل السطور بتاعت ال file وده بنستخدمه علشان ن print ال result

- **Records and Fields**

- في ال AWK by default اي Line بيقرءه بيسميه record وال record بيكون مقسم لمجموعه من ال Words وبيسميها Field وبناءا علي ال Field Separated(FS) وال default لل FS هو ال spaces او ال Tabs وال Linebreaks

$1	$2	$3	
abc	def	ghi	Line one
jkl	mno	pqr	Line two
stu	vwx	yz	Line three
Field 1	Field 2	Field 3	

- ف بيكون ال $0 بيكون شايل قيمه ال Line
- وال $1 بيكون شايل قيمه ال Field رقم 1 اللي هو العمود الاول اللي فيه abc و jkl و stu وهكذا
- و $2 هيكون شايل قيمه ال Filed رقم 2 وهكذا

- وال Record Number (NR) بيكون شايل رقم السطر اللي انت عليه
- ال Number of Field (NF) بيكون شايل عدد ال Field او ال Variable الموجوده في ال Line
- وال output record separator(ORS) وده بستخدمه لو انا عايز اتحكم في شكل الطباعه

- فيه عندي اوبشن في AWK بيخليني اغير في ال Field Separator وهو اوبشن F-

- يعني مثلا في ال Line ده ال AWK بيعتبر ال Line كله Field لان ال Field Separated(FS) ال default بتاعتها هي ال tab او ال space ف لو ملقاش اي مسافه هيعتبرها كلها field

```
root:x:0:0:root:/root:/bin/bash
```

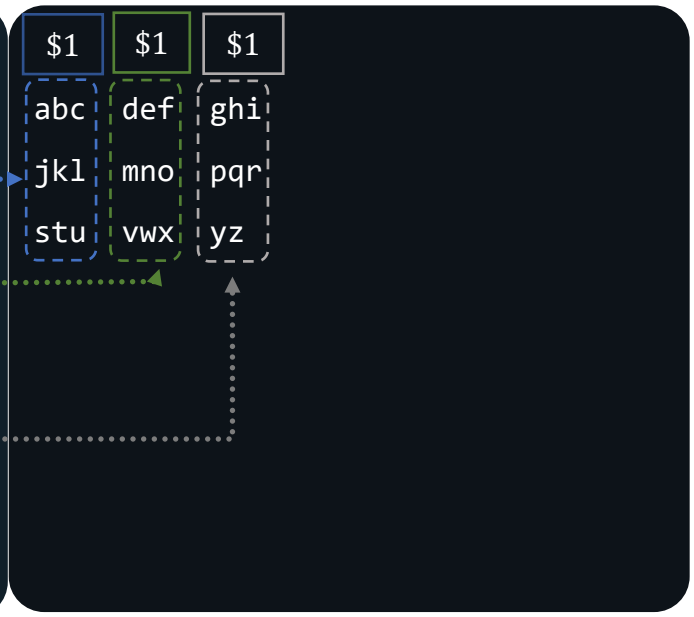
ده كده عباره عن Field

- لو انت عايز تغير ال Field Separator هتستخدم اوبشن F- وليكن مثلا انت عايز تغير الفاصل بدل اما كان مسافه تخليه مثلا : ف هتستخدم اوبشن **F:-** ف هو اول اما يشوف : هيعتبرها فاصل ف هيعتبر اللي قبلها field وهكذا وهنشوف ده بالامثله

```
root:x:0:0:root:/root:/bin/bash
```

ده كده كل جزء من ده عباره عن feild

```
MohamedAtef~$ awk '{ print $1 }'
  abc
  jkl
  stu
MohamedAtef~$ awk '{ print $2 }'
  def
  mno
  vwx
MohamedAtef~$ awk '{ print $3 }'
  ghi
  pqr
  yz
```



- ال file اللي هنشتغل عليه بيحتوي علي

```
mohamed@MohamedAtef:~$ cat > employee.txt
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

- لو احنا قولنا awk '{print}' employee.txt هو كده by default هيعمل Print لكل ال Lines اللي موجوده في الملف

```
mohamed@MohamedAtef:~$ awk   '{ print }'   employee.txt
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

- لو انا مثلا عايز اعرض كل ال Lines اللي بتحتوي علي ال Pattern اللي اسمه manager

```
mohamed@MohamedAtef:~$ awk '/manager/ { print }' employee.txt
ajay manager account 45000
varun manager sales 50000
amit manager account 47000
```

- لو انا عايز اقسم ال Line الي Field بمعني ان كل كلمه في السطر بتعتبر field بشرط ان يكون مابينها ومابين الكلمه اللي بعديها مسافه يعني اول كلمه بتعتبر field بتتخزن في $1 وتاني field بتتخزن في $2 وتالت field في $3 ورابع field في $4 وهكذا
- ف لو انت مثلا عايز تعرض كل ال Lines اللي بتحتوي علي ال Filed الاول والرابع هتستخدم

```
mohamed@MohamedAtef:~$ awk '{ print $1 , $4 }' employee.txt
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
deepak 23000
sunil 13000
```

انا هنا استخدمت مابينهم (,) comma علشان يفصل مابين ال Output ب space ولو انت مكتبتش ال comma هتلاقيه عمل concatenate مابينهم هتلاقيهم لازقين في بعض مفيش اي فاصل مابينهم هتلاقيهم كده ajay45000

- ممكن ان انت تستخدم ال NR علشان يعرضلك رقم السطر ال $0 بتحتوي علي محتوي ال Line

```
mohamed@MohamedAtef:~$ awk '{ print NR , $0 }' employee.txt
1 ajay manager account 45000
2 sunil clerk account 25000
3 varun manager sales 50000
4 amit manager account 47000
5 tarun peon sales 15000
6 deepak clerk sales 23000
7 sunil peon sales 13000
```

- ممكن كمان تستخدم ال $NF علشان تعرض اخر field

```
mohamed@MohamedAtef:~$ awk '{ print $NF }' employee.txt
45000
25000
50000
47000
15000
23000
13000
```

➢ لو انت عايز تعرض اول Field واخر Field هتستخدم $1,$NF

```
mohamed@MohamedAtef:~$ awk '{ print $1 , $NF }' employee.txt
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
```

➢ ممكن استخدم ال range ان انا عايز اعمل display من اول ال Field ال3 ل ال Field ال 6

```
mohamed@MohamedAtef:~$ awk 'NR==3, NR==6 { print NR , $0 }' employee.txt
3 varun manager sales 50000
4 amit manager account 47000
5 tarun peon sales 15000
6 deepak clerk sales 23000
```

- مثال تاني ومهم جدااا لان ده من اكتر الامثله اللي ممكن تقابلها وانت بتعمل Task او Project

➢ فيه command بنستخدمه دايما علشان نعرض بيه ال disk space usage وهو df -h

➢ ف هنعمل redirect لل output اللي ظهر ده ل file وليكن اسمه system.txt علشان نعرف نشتغل عليه بكل سهوله

```
mohamed@MohamedAtef:~$ df -h
Filesystem      Size   Used   Avail   Use%   Mounted on
tmpfs           782M   2.6M   779M    1%     /run
/dev/sdb3       219G   175G   33G     85%    /
tmpfs           3.9G   0      3.9G    0%     /dev/shm
tmpfs           5.0M   4.0K   5.0M    1%     /run/lock
tmpfs           3.9G   0      3.9G    0%     /run/qemu
/dev/sda5       98G    91G    6.9G    94%    /media/mohamed/DevOps
mohamed@MohamedAtef:~$ df -h > system.txt
```

➢ ف لو احنا عايزين نعمل Print لل field الاول وال field الرابع زي ماقولنا هنستخدم

```
$ awk '{ print $1, $4 }' system.txt
Filesystem      Avail
tmpfs           779M
/dev/sdb3       33G
tmpfs           3.9G
tmpfs           5.0M
tmpfs           3.9G
/dev/sda5       6.9G
```

$1	$2	$3	$4	$5	$6
Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	782M	2.6M	779M	1%	/run
/dev/sdb3	219G	175G	33G	85%	/
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	3.9G	0	3.9G	0%	/run/qemu
/dev/sda5	98G	91G	6.9G	94%	/media /DevOps

➢ ولو انت عايز تعرض رقم ال Line بس هتستخدم ال NR

```
$ awk '{ print NR }' system.txt
1
2
3
4
5
6
7
```

$1	$2	$3	$4	$5	$6
Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	782M	2.6M	779M	1%	/run
/dev/sdb3	219G	175G	33G	85%	/
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	3.9G	0	3.9G	0%	/run/qemu
/dev/sda5	98G	91G	6.9G	94%	/media /DevOps

➢ ولو انت عايز تعرض رقم ال Line مع مثلا ال field رقم 1 وال field رقم 4

```
$ awk '{ print NR $1 , $4 }' system.txt
1 Filesystem Avail
2 tmpfs 779M
3 /dev/sdb3 33G
4 tmpfs 3.9G
5 tmpfs 5.0M
6 tmpfs 3.9G
7 /dev/sda5 6.9G
```

$1	$2	$3	$4	$5	$6
Filesystem	Size	Used	Avail	Use%	Mounted on
tmpfs	782M	2.6M	779M	1%	/run
/dev/sdb3	219G	175G	33G	85%	/
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	3.9G	0	3.9G	0%	/run/qemu
/dev/sda5	98G	91G	6.9G	94%	/media /DevOps

➢ وممكن كمان تحددله انه يعرضلك من ال Line رقم7 مثلا ال filed رقم 4 اللي فيه النسبه ال Avail

```
$ awk 'NR == 7 { print $4 }' system.txt
6.9G
```

$1	$2	$3	$4	$5	$6
Filesystem	Size	Used	Avail	Use%	Mounted
tmpfs	782M	2.6M	779M	1%	/run
/dev/sdb3	219G	175G	33G	85%	/
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	3.9G	0	3.9G	0%	/run/qemu
/dev/sda5	98G	91G	6.9G	94%	/media /DevOps

➢ ممكن ان انت تستخدم ال Number of Field (NF) علشان تعرف عدد ال Field او ال Variable

```
$ awk { print NF }' system.txt
6 ده كده عدد ال Field الموجودين في اول سطر لو عددته هتلاقيهم
6 6 وهكذا بالنسبه للباقي
6
6
6
6
```

$1	$2	$3	$4	$5	$6
Filesystem	Size	Used	Avail	Use%	Mounted
tmpfs	782M	2.6M	779M	1%	/run
/dev/sdb3	219G	175G	33G	85%	/
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	3.9G	0	3.9G	0%	/run/qemu
/dev/sda5	98G	91G	6.9G	94%	/media /DevOps

➢ ممكن كمان ان احنا نعمل display لل Number of Field(NF) وال Record Number(NR)

```
$ awk { print NR , NF }' system.txt
1  6
2  6
3  6
4  6
5  6
6  6
7  6
```

```
$1          $2    $3    $4    $5    $6
Filesystem  Size  Used  Avail Use%  Mounted
tmpfs       782M  2.6M  779M  1%    /run
/dev/sdb3   219G  175G  33G   85%   /
tmpfs       3.9G  0     3.9G  0%    /dev/shm
tmpfs       5.0M  4.0K  5.0M  1%    /run/lock
tmpfs       3.9G  0     3.9G  0%    /run/qemu
/dev/sda5   98G   91G   6.9G  94%   /media /DevOps
```

➢ ممكن كمان ان احنا نعرض جمله بتوصف ال output وليكن احنا عايزين نعرض جمله Number of columns in line: ف علشان نعرض الجمله دي لازم نحطها مابين Curly Bracket { } وده معناه انه هينفذ الجمله دي مع كل سطر في ال AWK ولازم الجمله دي تتحط مابين " " Double Quotes

```
$ awk '{ print "Number of columns in line: " NR , "is " NF }' system.txt
Number of columns in line: 1 is 6
Number of columns in line: 2 is 6
Number of columns in line: 3 is 6
Number of columns in line: 4 is 6
Number of columns in line: 5 is 6
Number of columns in line: 6 is 6
Number of columns in line: 7 is 6
```

➢ ممكن كمان تستخدم ال $NF علشان تعرض اخر Field

```
$ awk { print $NF }' system.txt
Mounted
/run
/
/dev/shm
/run/lock
/run/qemu
/media /DevOps
```

```
$1          $2    $3    $4    $5    $6
Filesystem  Size  Used  Avail Use%  Mounted
tmpfs       782M  2.6M  779M  1%    /run
/dev/sdb3   219G  175G  33G   85%   /
tmpfs       3.9G  0     3.9G  0%    /dev/shm
tmpfs       5.0M  4.0K  5.0M  1%    /run/lock
tmpfs       3.9G  0     3.9G  0%    /run/qemu
/dev/sda5   98G   91G   6.9G  94%   /media /DevOps
```

- تعال نشوف ازاي نغير ال Field Separator وده عن طريق اوبشن F- وال F بتكون Capital
  - في الامثله اللي احنا شوفناها قبل كده كان ال Field Separator كان ال Space زي مثلا

```
mohamed@MohamedAtef:~$ df -h
Filesystem      Size    Used    Avail    Use%    Mounted on
tmpfs           782M    2.6M    779M     1%      /run
/dev/sdb3       219G    175G    33G      85%     /
tmpfs           3.9G    0       3.9G     0%      /dev/shm
tmpfs           5.0M    4.0K    5.0M     1%      /run/lock
tmpfs           3.9G    0       3.9G     0%      /run/qemu
/dev/sda5       98G     91G     6.9G     94%     /media/mohamed/DevOps
```

هنا كنا بنلاقي ان بين كل Filed والتاني او بين كل Variable والتاني
كان فيه مسافه ف كان اللي بيفصل ال Field عن التاني كان ال Space

- طب لو كان ال output بالشكل ده مكنش فيه اي space ال AWK هيعتبر ان كل ال Lines دي عباره عن Field واحد بس عمود واحد ف احنا هنغير بدل اما كان ال Field ال default بتاعه مسافه هنخليه مثلا : Colon

```
mohamed@MohamedAtef:~$ df -h
Filesystem:Size:Used:Avail:Use%:Mounted on
Tmpfs:782M:2.6M:779M:1%: /run
/dev/sdb3: 219G:175G:33G:85%: /
Tmpfs: 3.9G:0:3.9G:0%:/dev/shm
tmpfs:5.0M:4.0K:5.0M: 1%:/run/lock
tmpfs:3.9G: 0:3.9G :0%:/run/qemu
/dev/sda5:98G:91G:6.9G: 94%: /media/mohamed/DevOps
```

- ف احنا هنعمل file تاني وهيكون جواه محتوي المثال السابق وهنسمي ال file ده مثلا system1.txt ف احنا هنستخدم اوبشن F- علشان نغير ال Field Separator بدل اما كان space نخليه colon: ف احنا هنستخدم F- وبعدين مابين double quotes هتضيف ال : colon وبعدين في المثال احنا قايلين ليه انه يعمل display ل اول Field ف هنستخدم $1

```
$ awk -F ":" '{ print $1 }' system1.txt
Filesystem
Tmpfs
/dev/sdb3
Tmpfs
tmpfs
tmpfs
/dev/sda5
```

```
Filesystem:Size:Used:Avail:Use%:Mounted on
Tmpfs:782M:2.6M:779M:1%: /run
/dev/sdb3: 219G:175G:33G:85%: /
Tmpfs: 3.9G:0:3.9G:0%:/dev/shm
tmpfs:5.0M:4.0K:5.0M: 1%:/run/lock
tmpfs:3.9G: 0:3.9G :0%:/run/qemu
/dev/sda5:98G:91G:6.9G: 94%: /media/mohamed/DevOps
```

➢ ممكن ان احنا نستخدم ال cat وال | Pipeline مع ال AWK زي مثلا ان انا ممكن اقوله cat لل file اللي اسمه system1.txt واعرضلي منه ال field الاولاني

```
$ cat system1.txt | awk -F ":" '{ print $1 }'
Filesystem
Tmpfs
/dev/sdb3
Tmpfs
tmpfs
tmpfs
/dev/sda5
```

```
Filesystem:Size:Used:Avail:Use%:Mounted on
Tmpfs:782M:2.6M:779M:1%: /run
/dev/sdb3: 219G:175G:33G:85%: /
Tmpfs: 3.9G:0:3.9G:0%:/dev/shm
tmpfs:5.0M:4.0K:5.0M: 1%:/run/lock
tmpfs:3.9G: 0:3.9G :0%:/run/qemu
/dev/sda5:98G:91G:6.9G: 94%: /media/mohamed/DevOps
```

- هنشوف مثال تاني والمثال ده بيحتوي علي معلومات عن الموظفين من حيث الاسم واسم الاب والقسم والاميل والمرتب وهكذا وهيكون بين ال field والتاني | Pipeline ف هنشوف ازاي نغير ال Field Separator ل | ونشوف ازاي نستخدم ال Variable مع ال AWK

```
mohamed@MohamedAtef:~$ cat employees.txt
1|Mohamed|Atef|Finance|Financial Analyst|ahmed.hassan@company.com|60000
2|Nada|Hossam|IT|System Admin|Nada.Hossam@company.com|75000
3|Nourhan|Ashraf|IT|Software Developer|nourhan.ashraf@company.com|80000
4|Youssef|Kamal|Marketing|Marketing Specialist|youssef.kamal@company.com|65000
5|Hanan|Mahmoud|Sales|Sales Manager|hanan.mahmoud@company.com|90000
6|Omar|Sayed|IT|IT Manager|omar.sayed@company.com|95000
7|Sara|Amin|HR|HR Manager|sara.amin@company.com|85000
8|Mina|Naguib|Engineering|Engineering Manager|mina.naguib@company.com|100000
```

➢ احنا عايزين نعرض الجمله اللي هي Employee's First Name ويعرضلي ال field التاني ف احنا ممكن نعمل ده من خلال ال variable احنا استخدمنا ال Field Separator وهو | علشان ده الموجود مابين كل field مش ال Space وبعدين استخدمنا v- علشان بتعبر عن ال Variable وبعدين سمينا ال Variable بتاعنا var وخليناه بيحتوي علي جمله Employee's First Name: وبعدين قولنا ليه يعمل print لل var وال field التاني

```
$ awk -F "|" -v var="Employee's First Name: " '{ print var , $2 }' employees.txt
Employee's First Name: Mohamed
Employee's First Name: Nada
Employee's First Name: Nourhan
Employee's First Name: Youssef
Employee's First Name: Hanan
Employee's First Name: Omar
Employee's First Name: Sara
Employee's First Name: Mina
```

- ممكن ان احنا نستخدم ال condition مع ال AWK وبيكون مابين ' ' single quotes وهنا انت بتقوله اعرضلي كل ال lines او الموظفين اللي ال field رقم 7 فيها بيكون اكبرمن او يساوي 90000 اللي هو ال salary بتاعهم اكبر من او يساوي 90000 انا مستخدمتش هنا print علشان يعرضلي ال line كله

```
$ awk -F "|" '$7 >= 90000' employees.txt
5|Hanan|Mahmoud|Sales|Sales Manager|hanan.mahmoud@company.com|90000
6|Omar|Sayed|IT|IT Manager|omar.sayed@company.com|95000
8|Mina|Naguib|Engineering|Engineering Manager|mina.naguib@company.com|100000
```

- ممكن ان انت تستخدم ال Print عادي بالطريقه اللي موجوده في المثال واستخدمنا $0 علشان عارفين ان ال $0 بيعبر عن ال Line يعني بيعرضلي ال Line بالكامل

```
$ awk -F "|" '$7 >= 90000 { print $0 }' employees.txt
5|Hanan|Mahmoud|Sales|Sales Manager|hanan.mahmoud@company.com|90000
6|Omar|Sayed|IT|IT Manager|omar.sayed@company.com|95000
8|Mina|Naguib|Engineering|Engineering Manager|mina.naguib@company.com|100000
```

- وممكن ان احنا نستخدم ال Variable ان احنا نعرف Variable ونستخدمه في ال Condition

```
$ awk -F "|" -v high_salary="90000" '$7 >= high_salary' employees.txt
5|Hanan|Mahmoud|Sales|Sales Manager|hanan.mahmoud@company.com|90000
6|Omar|Sayed|IT|IT Manager|omar.sayed@company.com|95000
8|Mina|Naguib|Engineering|Engineering Manager|mina.naguib@company.com|100000
```

- ممكن ان احنا نستخدم اكتر من condition واكتر من variable لازم قبل ما اعرف اي Variable اضيف قبليه اوبشن v-

```
$ awk -F "|" -v high_salary="90000" -v low_salary="65000" '$7 >= high_salary || $7 <= low_salary { print $2 }' employees.txt
Mohamed
Youssef
Hanan
Omar
Mina
```

- هنا ضيفنا اكتر من variable واكتر من condation وكمان استخدمنا ال BEGIN

```
$ awk -F "|" -v high_salary=90000 -v low_salary=65000 -v header="=====salary header=====" 'BEGIN { print header } $7 >= high_salary || $7 <= low_salary { print $2 , $3 , $7 }' employees.txt
=====salary header=====
Mohamed Atef 60000
Youssef Kamal 65000
Hanan Mahmoud 90000
Omar Sayed 95000
Mina Naguib 100000
```

اكيد مش محتاجين شرح هنا 😂 الفكره في ال BEGIN زي ماقولنا قبل كده ان ال BEGIN هتتنفذ الاول قبل اما يقرأ ال file ف احنا عملنا Variable اسمه header وده بس عباره عن انه هيظهرلي في الاول ك توضيح للي هيتعرض وال BEGIN ده اوبشن ممكن استخدمه او او مستخدمهوش

Grep

- ال **Grep** وظيفته انه بيعمل search ب pattern معين جوه ال file ولو لاقي اي line بي match علي ال pattern ده بي display ال line اللي فيه ال Pattern ده

➢ تعال مثلا نعمل grep علي كلمه Mohamed في ال file اللي اسمه passwd اللي هيعمله ال grep انه هيعمل search علي ال pattern اللي اسمه Mohamed جوه ال file اللي اسمه passwd ولو لاقا ال pattern ده هيعرضه ويلون ال pattern ده

```
mohamed@MohamedAtef:~$ grep mohamed /etc/passwd
mohamed:x:1004:1005::/home/mohamed:/bin/bash
```

➢ هنعمل grep علي كلمه bash من ال /etc/passwd هنلاحظ انه ظهرلي كل ال lines اللي فيها كلمه bash

```
mohamed@MohamedAtef:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
Elbitawy:x:1000:1000:Elbitawy:/home/Elbitawy:/bin/bash
karim:x:1001:1001::/home/karim:/bin/bash
ansible:x:1002:1002::/home/ansible:/bin/bash
```

➢ ممكن استخدم اوبشن **-l** مع ال grep انه بدل اما يعرضلي ال lines اللي فيها ال pattern لا هو هيعرضلي ال file بمعني ان لو بعمل grep علي كلمه bash في ال /etc/passwd مش هيعرضلي ال lines زي مظهر في المثال اللي قبل ده لا هيعرضلي اسم ال file هو /etc/passwd كانه بيقولك اه موجود ال Pattern ده

```
mohamed@MohamedAtef:~$ grep -l bash /etc/passwd
/etc/passwd
```

➢ فيه اوبشن تاني ممكن استخدمه وهو **-i** بمعني case insensitive يعني بقوله اعمل search علي ال pattern سواء كان upper case او Lower case واعرضلي ال line ده

➢ لو انا جيت مثلا عملت grep علي كلمه ROOT وهي upper case في ال /etc/passwd مش هيعرضلي اي حاجه لان ال grep case sensitive

```
mohamed@MohamedAtef:~$ grep ROOT /etc/passwd
```

➢ لو نفذت نفس ال command واستخدمت معاه اوبشن **-i** هيعرضلي كل ال Line اللي فيها upper case و lower case

```
mohamed@MohamedAtef:~$ grep -i ROOT /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
```

- فيه اوبشن كمان وهو **-v** وده بستخدمه لو انا عايز اعرض كل ال lines ماعدا ال lines اللي فيها pattern معين

```
mohamed@MohamedAtef:~$ grep -v mohamed /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
```

- لو انا عايز اعمل سيرش بكلمه ذات نفسها هستخدم اوبشن **-w**

```
mohamed@MohamedAtef:~$ grep -w shut /etc/passwd
```

- لو انا عايز اعمل سيرش علي pattern معين وال Line اللي تلاقي فيه ال pattern ده اعرضه واعرض مثلا سطرين بعديه هستخدم اوبشن **-A** وبعدين العدد اللي انا عايز اعرضه

```
mohamed@MohamedAtef:~$ grep -A 2 root /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

- لو انا عايز اعمل العكس يعرضلي مثلا سطرين قبل ال pattern هستخدم اوبشن **-A** وبعدين عدد ال lines

```
mohamed@MohamedAtef:~$ grep -B 2 root /etc/passwd
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
```

- ممكن استخدم اوبشن **-r** علشان اعمل سيرش جوه ال directory بكل ال content بتاعته بال files بكل حاجه

```
mohamed@MohamedAtef:~$ grep -r mohamed /etc/
/etc/group:mohamed:x:1005:
/etc/gshadow:mohamed:!::
/etc/passwd:mohamed:x:1004:1005::/home/mohamed:/bin/bash
/etc/shadow:mohamed:$6$JZXFa4/ykqIK4fDK$BC2p4Cy.EWARMsQgZEKsv42I9DFQlk1cynbaHc7IBqPKymFBnt5ToY
8jHeKZ3UbvFOTXLhPRle6.uzltzxApI0:19912:0:99999:7:::
```

- ممكن اسخدم اوبشن **-lr** علشان اعرض اسامي ال files بدل اما يعرضلي ال Line اللي فيها ال pattern

```
mohamed@MohamedAtef:~$ grep -lr mohamed /etc/
/etc/group
/etc/gshadow
/etc/passwd
/etc/shadow
```

➢ لو انا عايز اعمل search ل اكتر من pattern في نفس الوقت هستخدم اوبشن **-e** مابين كلpattern

```
mohamed@MohamedAtef:~$ grep -e mohamed -e root /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:11:0:operator:/root:/sbin/nologin
mohamed:x:1004:1005::/home/mohamed:/bin/bash
```

دي كده بعض ال Regular Expressions اللي ممكن نستخدمها مع ال Grep

Symbol	Descriptions
.	replaces any character
^	matches start of string
$	matches end of string
*	matches up zero or more times the preceding character
\	Represent special characters
()	Groups regular expressions
?	Matches up exactly one character

- هنشوف ازاي نستخدم ال Regular Expressions مع ال Grep

- الدوت **.** بت match علي exactly one characters وبتستخدم مع ال grep command
- ال **?** Question mark بت matchعلي exactly one-character وبتستخدم مع ال ls command
- ال **^** دي بت Match علي start of string
- ال **$** dollar sign بت match علي end of string
- ال ***** معناها zero or more
- ال **** بت represent special characters
- ال **()** عباره عن Groups regular expressions

➢ لو انا عملت grep علي كلمه cat من ال file اللي اسمه words الموجود تحت /usr/share/dict/

```
mohamed@MohamedAtef:~$ grep cat /usr/share/dict/words
abacate
abboccato
abdicate
abdicated
```

ال file اللي اسمه words ده فيه تقريبا كل الكلمات الانجليزيه الموجوده عندي دا ال dictionary الموجود عندنا في اللينكس

➢ انا عايز أ list كل ال lines اللي بتبدء بكلمه cat او بال pattern ده هستخدم ال ^ في اول ال pattern

```
mohamed@MohamedAtef:~$ grep ^cat /usr/share/dict/words
cat
cat.
cata-
catabaptist
```

➢ لو انا عايز العكس ان انا أ list كل ال lines اللي بتنتهي بكلمه cat هستخدم $ في اخر ال pattern

```
mohamed@MohamedAtef:~$ grep cat$ /usr/share/dict/words
avocat
bearcat
blindcat
```

➢ طب لو انا عايز أ List كل ال lines اللي بتبدء بكلمه cat وبتنهي بكلمه cat ف انا هستخدم ال ^ في اول ال pattern وال $ في نهايه ال pattern

```
mohamed@MohamedAtef:~$ grep ^cat$ /usr/share/dict/words
cat
```

➢ ممكن استخدم الدوت . مع ال grep زي مثلا

```
mohamed@MohamedAtef:~$ grep c.t /usr/share/dict/words
abacate
abacot
abboccato
```

➢ ممكن اعمل search علي اي حاجه بتبدء ب c وبتنتهي ب t وفي النص a اوo او u

```
mohamed@MohamedAtef:~$ grep ^c[aou]t$ /usr/share/dict/words
cat
cot
cut
```

Variable

- الـ Variables بتبقى زي متغيرات في أي لغة برمجة، الهدف منها إنها تخزن بيانات نقدر نستخدمها بعد كده في الاسكربت.
- علشان نعرف Variables بنكتب اسم المتغير وبعده علامة = وبعدها القيمة اللي عايزين نخزنها. مفيش مسافات قبل وبعد علامه اليساوي = زي مثلا name="Mohamed" او age=25 وهكذا

```
mohamed@MohamedAtef:~$ name= "Mohamed"
mohamed@MohamedAtef:~$ age=25
```

- علشان نطبع القيمه الموجوده في ال name بنحط $ قبل اسم المتغير علشان يعرف ان ده عباره عن Variable ولو عايز اطبع القيمه هستخدم echo $name

```
mohamed@MohamedAtef:~$ echo $name
Mohamed
```

- ولو انت مستخدمتش ال dollar sign $ هيطبعلك ال name كانه اسم

```
mohamed@MohamedAtef:~$ echo name
name
```

- طب لو انا استخدمت مع ال variable ال Double Quotation هل هيعرضلي ال username ولا هيعمل Ignore هيعرضلي ال Variable وهيعمل enable لان احنا قولنا انه بيعمل quoting لبعض ال special characters وفيه بعض الحاجات زي ال Command Substitution وزي ال Variable Substitution بيسمحلك بيها

```
mohamed@MohamedAtef:~$ echo "$name"
Mohamed
```

- ولو استخدمنا نفس المثال بس مع ال Single Quotationهيعمل ignore لل dollar sign($) وهيعرضلي نفس ال command

```
mohamed@MohamedAtef:~$ echo '$name'
$name
```

- لو جينا عملنا x=5 و y=6 وجينا عملنا z=x+y هل لو احنا طبعنا قيمه ال z هيطلعلنا مجموعهم ولا ايه بالظبط هيطبعلنا x+y لانه فهم ان ال x+y عباره عن String

```
mohamed@MohamedAtef:~$ x=5
mohamed@MohamedAtef:~$ y=6
mohamed@MohamedAtef:~$ z=x+y
mohamed@MohamedAtef:~$ echo $z
x+y
```

- ولو احنا عملنا z=$x+$y هيطبعلي قيمه 5+6 ك text

```
mohamed@MohamedAtef:~$ echo $z
5+6
```

- علشان نحل المشكله دي عندي اكتر من طريقه اول طريقه ان وانت بتعرف ال z لازم تعمل declare -i ال -i يعني Integer واقوله ان z=$x+$y ولو عملنا echo $z هيطلعنا قيمه ال z ب 11

```
mohamed@MohamedAtef:~$ declare -i  z=$x+$y
mohamed@MohamedAtef:~$ echo $z
11
```

- تاني طريقه ودي الافضل في الاستخدام ان انا اقوله z وبعدين يساوي وبعدين $ dollar sign وبين قوسين زي كده (( )) بضيف جواهم $x+$y

```
mohamed@MohamedAtef:~$ z=$(($x+$y))
mohamed@MohamedAtef:~$ echo $z
11
```

- تالت طريقه ان انا ممكن استخدم ال let عن طريق ان انا هقوله let z=$x+$y

```
mohamed@MohamedAtef:~$ let z=$x+$y
mohamed@MohamedAtef:~$ echo $z
11
```

- رابع طريقه ان انا ممكن استخدم ال typeset -i

```
mohamed@MohamedAtef:~$ typeset -i z=$x+$y
mohamed@MohamedAtef:~$ echo $z
11
```

- سؤال مختلف هل ال Variable دي بتكون موجوده فين او بتتحفظ فين وهل فعلا بتكون موجوده ممكن ان احنا نستخدم مثلا command اسمه set ونعمل grep علي ال Variable ونشوف هل هو موجود ولا لا
- لو جينا عرفنا variable اسمه Mohamed_Atef وخليناه بيساوي 26 وعايزين نعرف هل ال Mohamed ده موجود فعلا ولا لا ف هنستخدم ال set مع ال grep ونشوف

```
mohamed@MohamedAtef:~$ mohamed_Atef=26
mohamed@MohamedAtef:~$ set | grep "mohamed_Atef"
mohamed_Atef=26
```

- لو انا عندي Variable وعايز امسحه بعمل unset لل Variable

```
mohamed@MohamedAtef:~$ unset mohamed_Atef
```

- **Environment Variables**
- الأمر export في الشيل بيُستخدم لتحويل ال Shell Variable إلى Environment Variables، بحيث تكون متاحة ليس فقط في ال Shell الحالية، لكن كمان لأي عمليات أو برامج تشتغل في نفس ال Shell.
- بمعنى تاني، لما تعمل export ل Variable، ال Variable ده هيبقى متاح في كل العمليات الفرعية اللي تبدأ من ال Shell دي. العمليات دي ممكن تكون أسكربتات شيل أو برامج تانية.

```
mohamed@MohamedAtef:~$ export mohamed_Atef=26
mohamed@MohamedAtef:~$ echo $mohamed_Atef
26
mohamed@MohamedAtef:~$ bash
mohamed@MohamedAtef:~$ echo $mohamed_Atef
26
```

لو احنا عملنا export ل variable اسمه Mohamed_Atef وعملنا echo ليه هيطبعلنا القيمه بتاعته ولو فتحنا shell تانيه باستخدام command اسمه bash وعملنا echo لنفس ال variable هيطبعلنا ال قيمه ال variable

- ف ال export بيورث قيمه ال Variable لباقي الشيل

Shell Scripting

BY: Mohamed Atef Elbitawy

https://www.linkedin.com/in/mohamedelbitawy

https://github.com/MohamedAtefElbitawy