**Bash Shell Scripts-**

**Wildcards:**

Working under bash.

|  |  |
| --- | --- |
| משמעויות | סימנים |
| **כל תו (לא קבצים נסתרים)** | **\*** |
| **כל תו בודד (לא ריק)** | **?** |
| **כל תו המופיע ברשימה (תו בודד)** | **[ab\*], [0-9], [a-z A-Z 0-9]** |
| **כל האותיות קטן או גדול (תו בודד)** | **[[:lower:][:upper:]]** |
| **כל תו שאינו מופיע ברשימה (תו בודד)** | **[!a3j] = [^a3j]** |

Man 7 glob

**Input redirection:**

Set -o noclobber - to prevent overwrite {a > list}

Set +o noclobber - disable set -o

noclobber - continues until the end of the session

If I add pipe after > I can overwrite {a>| list} even if set -o enable

Set -o - show the commands status

cat > z -- send output to file z without error

cat 1> z - send all output including errors

cat 2> z -- send error output to z

**Grep:**

-w - specific word

-I - list

-v - show who is not

-c - count grep selections

-e - expression

-i - ignore case

-R - recursive

-A - after[num]

-B - before[num]

**SED**

sed ­nei [line id] [sed command] [input] (line id ­ or num or regex)

­n ­ silent mode

­e can run multiple sed commands spaced by; starts and ends with ‘

­i insert changes

­nei [line id1, line id2] [sed command] [input]

Runs sed command on all lines between [line id1] to [line id2]

Sed commands:

p ­ print (prints the all-relevant lines)

d ­ delete (deletes all the relevant lines)

a txt ­ (appends the txt under the relevant lines)

i txt ­ (inserts the txt above the relevant lines)

w file ­ (writes the relevant lines to file)

s /old expression/new expression/[gi] ­ (replaces the old to new in relevant lines)

**AWK**

Built in vars:

NF ­ num of rows or last row

NR ­ num of rows processed

FNR ­ num of lines in the file

FS ­ Field separator

RS ­ Record separator

OFS ­ Output field separator

ORS - Output Record separator

**Find**

0 - try to find only in the folders that I listed without the folder list

1. folder list without sub folders
2. all folders including sub folders

-name (wildcards) - file name

-user - owner

-type - file type (f,d,l,b,c,p,s)

inum - inumber

-size - file size (1M)

-mtime - (modified files days - 0,-7)

-mmin - (modified files minute - 10,-60)

-o = OR

example: *find /home /tmp -maxdepth 2 -user $(whoami) -o*

*-user root -name “\*.log”* - **try to find all logs files in 2 subfolders maximum with the user owner root and me**

-exec {} \; - take the output and running program on it, example: -exec *rm -f {} \; -* delete the file we just found

-exec=-ok

**Column filters:**

cut -c 1-3 - show the first 3 letters

cut -d : -f 1,3 – show field 1,3 and : between

awk -F (space) '{print "text",$1,NF}'

NF - how many columns in the line

$NF - the last column in the line

$NF-1 - one column before the last one

$0 - all the line

[.] - Present point

\. - present point

**MISC filters:**

tr r t - replace r with t

tr -d - deleted letter

sort - (by default alphabetically) [­n num] [­r reverse]

[­k5 field 5] [­-g sort small to big] [­t מפריד תו]

uniq - show Identity lines

grep $(whoami) /etc/passwd - take the output of whoami (current user) and search it in the file

/etc/passwd | sort -nk3 -t: -- sort follow the 3rd column and sort them with:

**Command Substitution:**

CMD1 $(CMD2 $(CMD3))

**Regular Expressions**

Grep**:**

|  |  |  |
| --- | --- | --- |
| **בשורה מיקום** | **כופלים** | **תווים** |
| ^ 🡪 begin line | [0-9]\+ 🡪 show **all** the numbers in the world | . 🡪 any sign without /n |
| $ 🡪 end of line | [0-9]\{9\} 🡪 show 9 numbers in a row (like id.) | [ab.], [0-9], [a-z A-Z 0-9] |
| ˆa…..z$ 🡪 begin and end of line | [0-9]\{3,10\} 🡪 show the row numbers 3 to 10 ((123 - 1234567890)) | [^a-k] - not a till k |
|  | \+$ 🡪 **have** to be letter after the sign |  |
|  | {\5,\} 🡪 means that I will grep 5 in a row endlessly |  |

Grep “ˆvi\(able\)\*$”🡪 means that grep try to find **vi** and **viable** 🡪 vi, viable

Grep “ˆviable\*$” 🡪 means that **e** is optional 🡪 viable, viabl (without **e**\)

Grep “c$” - end with c

**Bash Mechanisms**

Variables - cell in memory with value - setting up vars; var=value

*set -* show the variables in the session

*env -* show the exported variables

$var - shown variable

alias rm=rm -I (alias for rm -i)

**Environment files:**

|  |  |  |
| --- | --- | --- |
|  | Global | Personal |
| Login: | /etc/profile, /etc/profile.d | ~/.bash\_profile |
| Shell: | /etc/.bashrc | ~/.bashrc |
| Logout: |  | ~/.bash\_logout |

Execution order (login order)

/etc/profile 🡪 ~/.bash\_profile 🡪 ~/.bashrc 🡪 /etc/bashrc 🡪 ~/.bashrc 🡪 ~/.bash\_profile

System variables = $

UPPER CASE IMPORTANT!

Alias, vars, env, path

PATH 🡪 folder:/folder/folder:/folder

**How to write script:**

File type - .sh

Start with - *#!/bin/bash*

# - comment

echo $\* - will treat (show) any argument I sent outside the script

echo $0 - script name

echo $# - counting the arguments

echo $@ - all the arguments as a string

echo $2 - shows the second arguments

echo ${10} - shows the 10th argument

echo ${$#}

echo $$ - PID

echo ${!#} - shows the last argument

read -p - ask me to enter argument “enter you name: “ **name**

$? - exit status (shows me the last command error status):

0 - no errors

1 + 2 - errors

**Example:**

read -p "what is your chose: " choice

owner=$(ls -ld $1 | awk '{print $3}')

size=$(ls -l “$1” | awk ‘{print $5}’)

echo “the size of $1 is $size bytes

**takes the file and read the information, column 5 is the file size so the output will be “the size of *file1* is 46 bytes”**

**how to run script?**

|  |  |  |
| --- | --- | --- |
| Running command | Procedure path | Permissions |
| Bash /path/script.sh  Bash script.sh  -v -verbose (will show the full commands + the output)  -x - debug (will show the commands with separation between output to commands) | Bash🡪scrip🡪bash  (If I will run the command out the script on bash env. The variables I configured inside the script will not be available to use) | r (read) |
| Full path: /path/script.sh  Relative path: ./script/sh | Bash🡪scrip🡪bash  (If I will run the command out the script on bash env. The variables I configured inside the script will not be available to use) | rx (read and execute) |
| . /path/script.sh | Bash🡪scrip🡪bash  (If I will run the command out the script on bash env. There variables I configured inside the script will be available to use) | r (read) |

**Conditions:**

***Conditions of File:***

*[-e file1] - if the file exist* -ot/nt – older/newer then

*-f,-d,-L,-r,-w,-x – if is file,dir,link,read,write,execute*

*[file1 -nt file2] - if the file is* ***n****ewer* ***t****hen file 2*

**Conditions on Strings:**

[-z string] - if the string is 0 [!-z = -n] – if the string not 0

file1 = file2 - if the file1 string is equal to file2

< , > , = , <=, != not equal, =~ similar or equal [a\*=~abc]

**Conditions Numbers:**

-ef - like hard link -ne/eq - not/equal

-lt - less then < -gt - greater then >

-le - less equal -ge - greater equal

**Conditions Execution:**

$> *CMD1* && *CMD2* --> CMD2 will execute only if CMD1 succeeded ($?=0)

$> *CMD1* || *CMD2* --> CMD2 will execute only if CMD1 failed ($?≠0)

*1.*

*if [ $stat -eq 0 ] is condition*

*then*

*(command)*

*elif [ $stat -eq 5 ]*

*then*

*(command)*

*fi*

*2.*

*if (conditions) is equal to one command*

*then*

*(commands)*

*else*

*(commands)*

*fi*

*3.*

*if (conditions)&& (conditions) like "and" equal 2 commands*

*then*

*(commands)*

*elif (conditions)|| (conditions) like "or"*

*then*

*(commands)*

*fi*

***NOTE:***

***If HAVE to be with [space command space]***

***Loops:***

* *While*

*While condition*

*Do*

*Command*

*Done*

*While [$# -gt 0] (in condition the number of the items is greater then 0)*

*do*

*echo $1(print the 1st number)*

*shift (do not repeat after finish one)*

*done*

*cat file | while read my-line (read from file and take the output and paste it to the read command in the cell "my line")*

*do*

*echo $my-line*

*done*

* *For*

for var [in value list]

do

command

done

for fruit in mango apple orange (for group **fruit** add the fruits list)

do

echo "I like ${fruit}'s" (print the item inside the output)

done

for dir in "$1"/\* (inside the dir add all the content of the value (\* - wildcard)

do

[ -d "$dir ] && ls -ld "$dir" (then if the output is a directory then do ls -ld to read all the content)

done

for file in /home/\*/.bashrc; do sed a txt $file (……)

Script to read from file and check if he exists in the system

#!/bin/bash  
  
cat file | while read x (read external file with names one by one and add it to **x**)  
do  
if  [ "$(cat /etc/passwd | grep ^$x | awk -F: '{print $1}')" = "$x"] (we checking in /etc/passwd file all the lines that starting with one of the names that written in **x** and if the name is equal to one of the names in **x** so we "success", otherwise we "finish")   
then  
        echo "success"  
        else  
        echo "finish"  
        fi  
done

package-cleanup --oldkernels --count=1 מוחק את האפשרויות הישנות

conf /etc/vim/vimrc-" Set scripts to be executable from the shell

au BufWritePost \* if getline(1) =~ "^#!" | if getline(1) =~ "/bin/" | silent !chmod +x <afile> | endif | endif