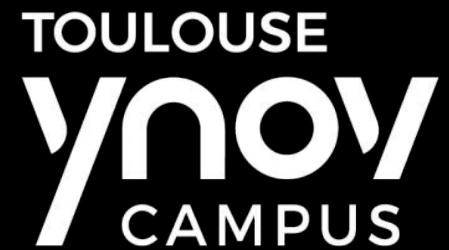


Scripting

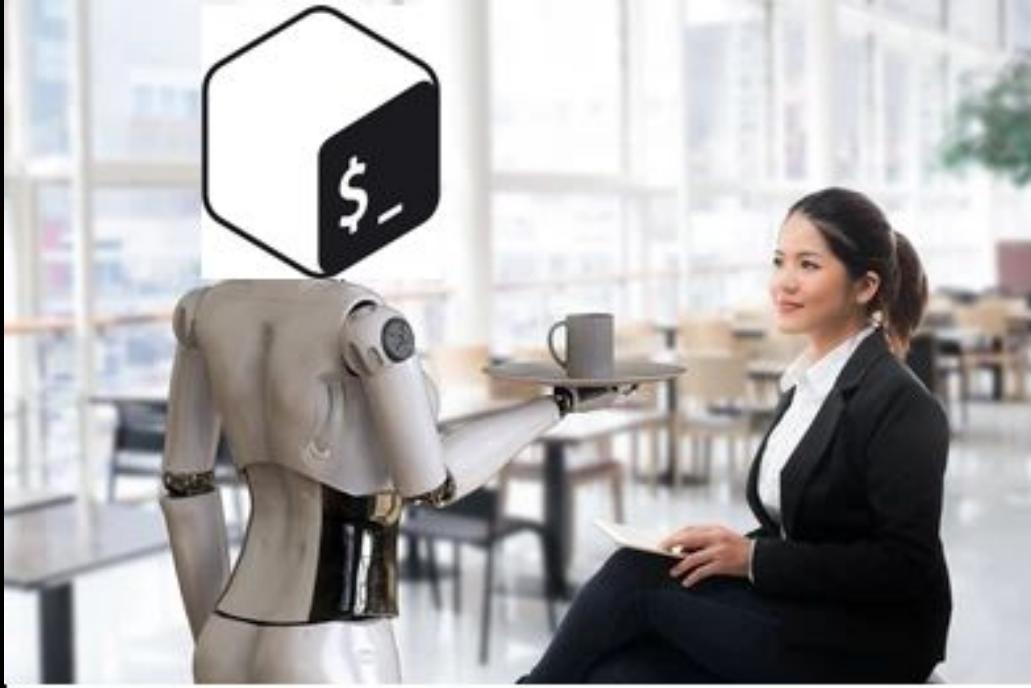


Scripting?

Le scripting consiste à coder pour automatiser des processus fastidieux à faire manuellement



Objectif du cours



Scripting?

- Langage “haut niveau”

Scripting?

- Langage “haut niveau”
- Langage interprété

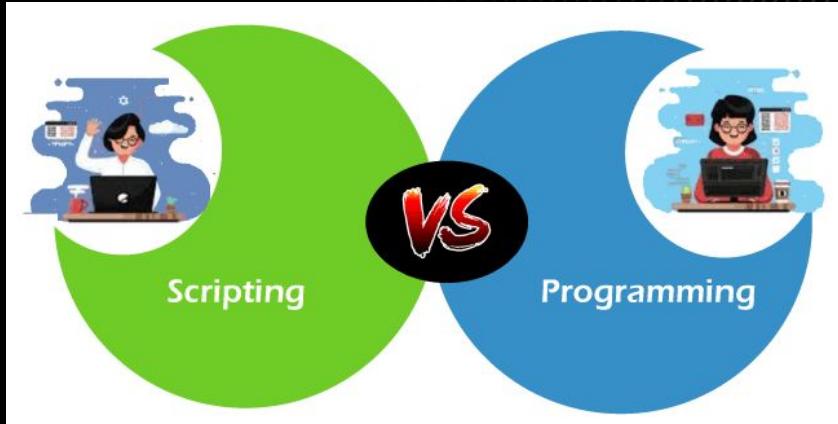
Scripting?

- Langage “haut niveau”
- Langage interprété
- Pas de compilation

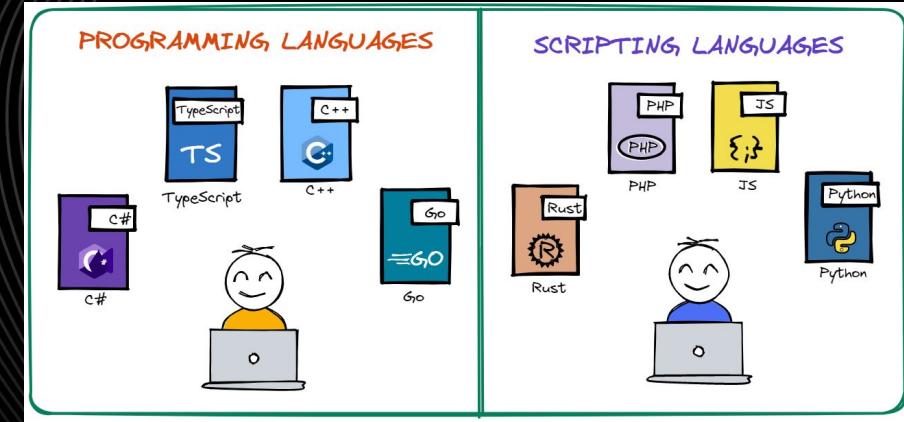
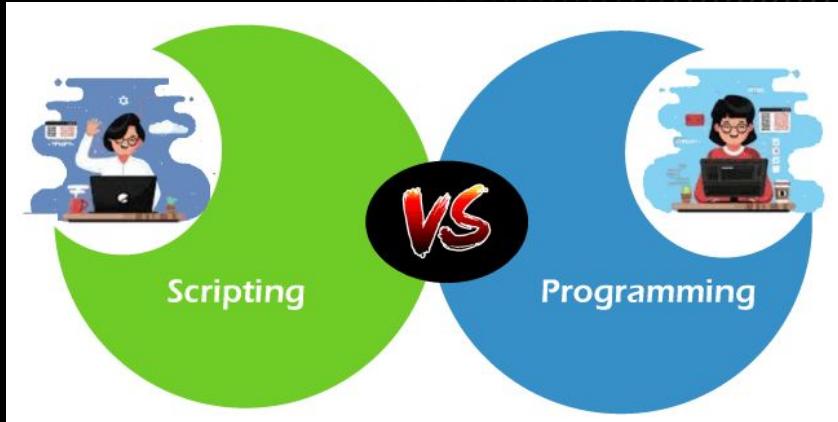
Scripting?

- Langage “haut niveau”
- Langage interprété
- Langage non typé
- Pas de compilation
- Utile pour combiner des composants déjà existants

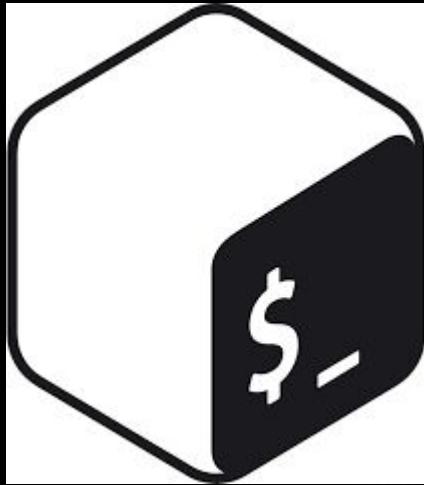
Scripting?



Scripting?

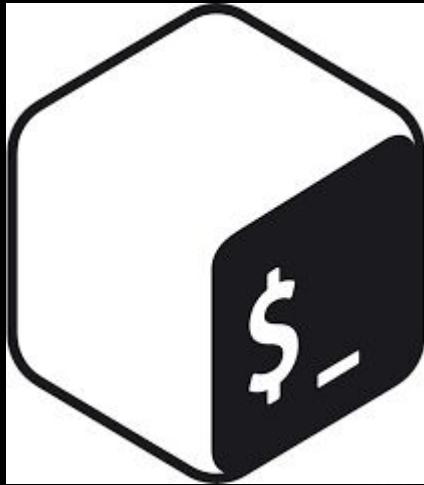


Bash?



Bourne again shell

Bash?



Bourne again shell

Terminal & language

Pourquoi Bash?

- Accéder à l'OS de manière performante
- Automatiser des tâches
- Lancer n'importe quel langage de prog
- Utiliser n'importe quel outil (BDD, git, docker...)
- Popularité de bash (serveurs linux, support de bash, communauté)

Comment utiliser Bash?



WSL 2



The Microsoft Store page for WSL 2 features a large blue banner with the text "Run Linux on Windows" and "Install and run Linux distributions side-by-side on the Windows Subsystem for Linux (WSL)". It includes a small image of a penguin and a screenshot of a terminal window. Below the banner, there are five cards representing different Linux distributions:

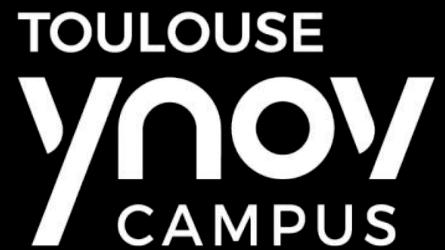
Distribution	Status	Rating
Ubuntu	Installed	★★★★★
openSUSE Leap 42	Installed	★★★★☆
SUSE Linux Enterprise Server 12	Owned	★★★★☆
Debian GNU/Linux	Installed	★★★★☆
Kali Linux	Owned	★★★★★

Comment utiliser Bash?



Visual Studio Code

Maîtriser les commandes linux



Step 1



.-/+o0ssssso0t/- .
`:+ssssssssssssssssssssssssss+`
-+ssssssssssssssssssssssssssyyssss+-
.osssssssssssssssssdmmNyssso.
/sssssssssssshdmmNnmmyNMmmMhssssss/
+sssssssssshydMMmmMMMddddyssssssss+
/sssssssssshnmmMyhyyyyhdnmMMNhssssssss/
.ssssssssssdmMMNhsssssssssshhNMmMdssssssss.
+sssssshhhyNMmNyssssssssssssyNMmMyssssss+
ossyNMmMyNMhsssssssssssshhmmmhssssssssso
ossyNMmMyNMhssssssssssssshmmmhssssssssso
+sssssshhhyNMmNyssssssssssssyNMmMyssssss+
.ssssssssssdmMMNhsssssssssshhNMmMdssssssss.
/sssssssssshnmmMyhyyyyhdnmMMNhssssssss/
+ssssssssssdmydMMmmMMMddddyssssssss+
/sssssssssssshdmmNnmmyNMmmMhssssss/
.osssssssssssssssssssdmmNyssso.

bajcmartinez@xps-linux

```
OS: Ubuntu 20.04.1 LTS x86_64
Host: XPS 15 9570
Kernel: 5.4.0-48-generic
Uptime: 12 days, 21 hours, 48 mins
Packages: 2117 (dpkg), 13 (snap)
Shell: zsh 5.8
Resolution: 3840x2160
DE: Plasma
WM: KWin
Theme: Breeze [Plasma], Breeze [GTK2/3]
Icons: breeze [Plasma], breeze [GTK2/3]
Terminal: konsole
CPU: Intel i7-8750H (12) @ 4.100GHz
GPU: Intel UHD Graphics 630
GPU: NVIDIA GeForce GTX 1050 Ti Mobile
```

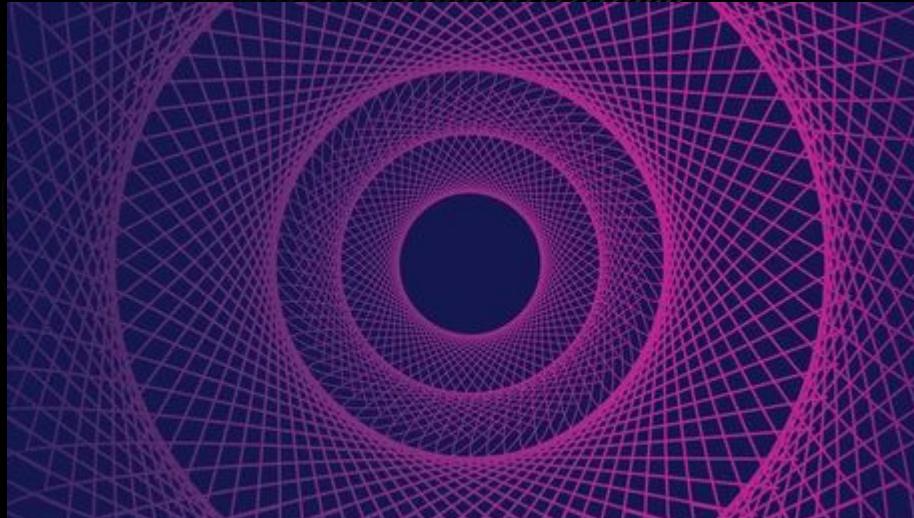
Rappels des commandes linux

```
fuoko@fuoko: ~
```



The ASCII art depicts a figure standing on a path made of dashed lines. The figure's body is formed by brackets and parentheses, with a large 'D' for a head. It wears a 'H' shaped hat and a 'I' shaped necktie. The figure is holding a 'Y' shaped object in its hand. The background features a path leading towards a 'A' shaped tree.

Rappels des commandes linux



echo “hello world!”

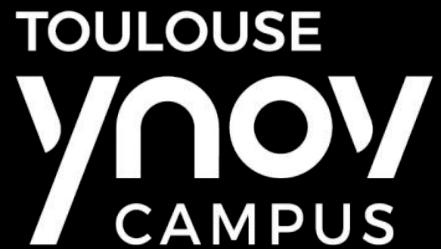
Rappels des commandes linux

File commands	
ls	Directory listing
ls -al	Formatted listing with hidden files
cd dir	Change directory to dir
cd	Change to home
pwd	Show current directory
mkdir dir	Create a directory dir
rm file	Delete file
rm -r dir	Delete directory dir
rm -f file	Force remove file
rm -rf dir	For remove directory dir
cp file1 file2	Copy file1 to file2
cp -r dir1 dir2	Copy dir1 to dir2; create dir2 if it doesn't exist
mv file1 file2	Rename or move file1 to file2. If file2 is an existing directory, moves file1 into directory file2
ln -s file link	Create symbolic link link to file
touch file	Create or update file
cat > file	Places standard input into file
more file	Output the contents of file
head file	Output the first 10 lines of file
tail file	Output the last 10 lines of file
tail -f file	Output the contents of file as it grows, starting with the last 10 lines

Exercice 1

- 1) Dans les documents, créer un répertoire exo_noob
- 2) Dans ce répertoire, créer un **répertoire** exo1 et un **fichier** exo1
- 3) Editer le fichier exo1 avec la commande nano et écrire comme contenu “ezpzlifeisez”
- 4) Afficher le contenu du fichier exo 1 à l'aide d'une commande
- 5) Renommer le répertoire exo1 en exo_facile avec une commande
- 6) Supprimer le répertoire exo_noob avec une commande

Les outils stylés



Pipe |

Transfère l'output d'une commande à l'input de la suivante



COMMANDÉ 1

|



COMMANDÉ 2

Pipe |

Transfère l'output d'une commande à l'input de la suivante

LS



WC -L

COMMANDÉ 1

COMMANDÉ 2

Pipe |

Transfère l'output d'une commande à
l'input de la suivante

HISTORY

GREP LS

COMMANDÉ 1

COMMANDÉ 2

Redirection sorties > et >>



```
ls > list_files.txt  
echo "lut" > sa
```

Opérateur wildcard

*



Opérateur wildcard

*



Remplacé par n'importe quelle
chaîne de caractère

Opérateur wildcard

*

```
cp * /other/path
```

```
rm *
```

```
mv *.txt /other/path
```



Opérateur wildcard

?



Opérateur wildcard

?



Remplacé par n'importe quel caractère unique

Opérateur wildcard

?

cp ??? /other/path



Plus de commandes !

tree

find [path] -name [motif]

find ./ -name fichier

find ./ -name fich*

Exercice 2

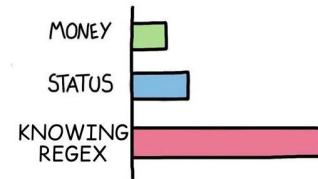
- 1) Dans dossier, créer 5 fichiers
 - one (contenu 1)
 - two (contenu 2)
 - three (contenu 3)
 - four (contenu 4)
 - five (contenu 5)
- 2) Afficher le contenu des fichiers qui ont un nom de trois lettres avec le wildcard ?
- 3) Afficher le contenu des fichiers qui commencent par la lettre t avec le wilcard *
- 4) Ecrire le contenu des cinq fichiers dans un nouveau fichier fusion.txt (redirection)
- 5) Ecrire une commande qui affiche le nombre de fichiers dans ce répertoire

Expressions régulières

The chances of me understanding
REGEX



WHAT GIVES PEOPLE
FEELINGS OF POWER



Expressions régulières

example@gmail.com

([a-zA-Z0-9_.+]+)@[a-zA-Z0-9_.+]+\\.[a-zA-Z0-9_.+]

Expressions régulières

Applications :

Extractions de données

Search & replace

Scraping

Versatile

...

Découverte de grep et des expressions régulières

Character Classes		Quantifiers		Anchors
[abc]	A single character of: a, b or c	a?	Zero or one of a	\G
[^abc]	A character except: a, b or c	a*	Zero or more of a	Start of match
[a-z]	A character in the range: a-z	a+	One or more of a	^
[^a-z]	A character not in the range: a-z	[0-9]+	One or more of 0-9	Start of string
[0-9]	A digit in the range: 0-9	a{3}	Exactly 3 of a	\$
[a-zA-Z]	A character in the range: a-z or A-Z	a{3,}	3 or more of a	\A
[a-zA-Z0-9]	A character in the range: a-z, A-Z or 0-9	a{3,6}	Between 3 and 6 of a	Start of string
		a*	Greedy quantifier	\Z
		a*?	Lazy quantifier	End of string
		a*+	Possessive quantifier	\z
				Absolute end of string
				\b
				A word boundary
				\B
				Non-word boundary

Regex101.com, site pour tester ses expressions

The screenshot shows the Regex101.com web application interface. The top navigation bar includes links for @regex101, \$ donate, ❤ sponsor, 📧 contact, ⚠ bug reports & feedback, 📖 wiki, and 🌐 what's new. The main workspace is divided into several sections:

- REGULAR EXPRESSION:** A text input field containing the placeholder text "i / insert your regular expression here" with a "no match" status indicator.
- TEST STRING:** A large text input field with the placeholder text "Insert your test string here".
- EXPLANATION:** A section stating "An explanation of your regex will be automatically generated as you type."
- MATCH INFORMATION:** A section stating "Detailed match information will be displayed here automatically."
- QUICK REFERENCE:** A table with two rows:
 - Search reference: A single character of: a, b or c [abc]
 - All Tokens: A character except: a, b or c [^abc]

On the left side, there is a sidebar with various tools and settings:

- SAVE & SHARE:** Includes "Save Regex" (ctrl+s).
- FLAVOR:** Set to PCRE2 (PHP >=7.3). Other options include PCRE (PHP <7.3), ECMAScript (JavaScript), Python, Golang, Java 8, and .NET (C#).
- FUNCTION:** Set to Match. Other options include Substitution, List, and Unit Tests.
- TOOLS:** Includes Code Generator and Regex Debugger.

Découverte de grep et des expressions régulières

```
grep bonjour bonjour.txt
```

```
grep -E <regex> bonjour.txt
```

```
grep -oE <regex> bonjour.txt  
(-o garde seulement l'expression matchée)
```

Découverte de grep et des expressions régulières

MOT QUI COMMENCENT PAR COR

CORDE
CORSE
CORNE

CONTE
COMME
COLIBRI
ACCORD

Découverte de grep et des expressions régulières

MOTS QUI COMMENCENT PAR BA ET FINISSENT PAR E

BASSE
BACTERIE
BAIGNOIRE

BONJOUR
BILLE
EBahi
BASSIN

Découverte de grep et des expressions régulières

MOTS DE CINQ LETTRES

BOULE
BONTE
ALLEZ

ALLO
INCROYABLE
MAISON

Découverte de grep et des expressions régulières
MOTS QUI CONTIENNENT QUE DES
VOYELLES

EAU
OUI
OU

AILE
SOU

Découverte de grep et des expressions régulières

MOTS QUI ALTERNENT SUCCESSIVEMENT CONSONNES ET VOYELLES

PATATE

SOPALIN

MENAGE

EAU

BONJOUR

COULEUR



CURL <URL>

Exercice Regex

- 1) Créer un répertoire exo2 dans le dossier personnel
- 2) Télécharger le zip du dico à cette adresse (wget)
http://www.3zsoftware.com/listes/liste_francais.zip
- 3) Installer le paquet unzip avec apt
- 4) Utiliser la commande unzip pour extraire le fichier du zip
- 5) supprimer liste_francais.zip avec rm
- 6) renommer liste_francais.txt en dico.txt avec mv
- 7) Ecrire tous les mots qui commencent par z et qui finissent par e dans un fichier z-e.txt
- 8) Afficher le nombre de mots dans ce fichier z-e.txt avec une commande

BONUS :

Afficher le nombre de commandes grep que vous avez utilisé depuis le début !

**Récupérer avec une requête curl seulement
les pays qui commencent par “Al”**

Utiliser ce site :

<https://www.atlas-monde.net/tous-les-pays/>



Scripting



bash
saying my
script has
errors

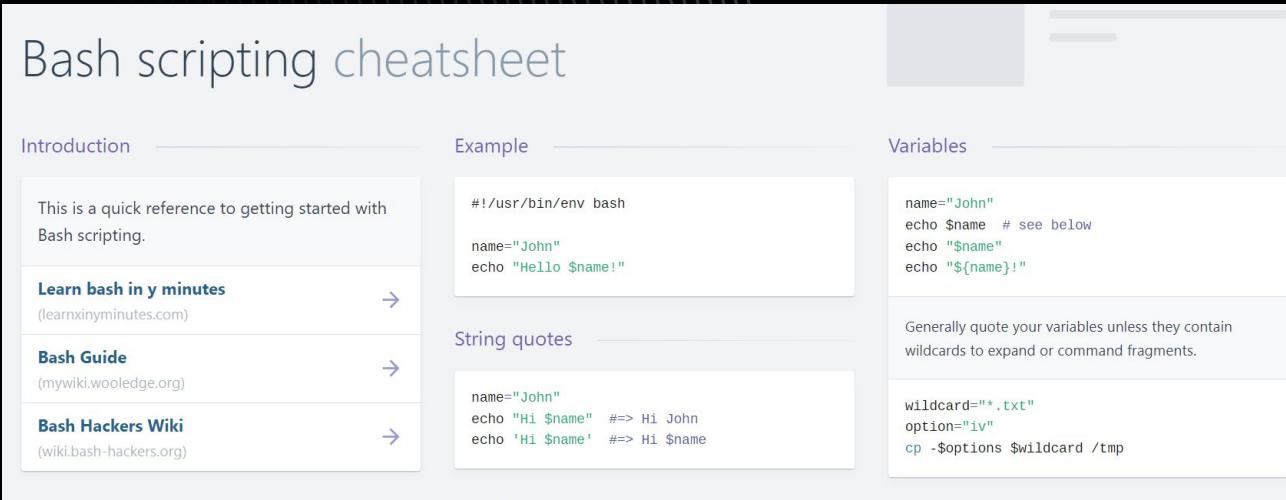
me who can't even write a
single line of bash without
consulting google



Cheat sheets

<https://devhints.io/bash>

<https://devhints.io/regexp>



The screenshot shows a cheatsheet for Bash scripting. It has a header "Bash scripting cheatsheet" and three main sections: "Introduction", "Example", and "Variables".

- Introduction:** This is a quick reference to getting started with Bash scripting.
- Example:** A code example showing how to use variables and string quotes.

```
#!/usr/bin/env bash

name="John"
echo "Hello $name!"
```
- Variables:** A code example showing variable assignment and expansion.

```
name="John"
echo $name # see below
echo ${name}
echo ${name}!"
```

String quotes

```
name="John"
echo "Hi $name" #=> Hi John
echo 'Hi $name' #=> Hi $name
```

SCRIPTS BASH

GNU nano 2.9.3

script.sh

```
#!/bin/bash
mkdir ./dossier1
cd ./dossier1
touch fichier1
touch fichier2
```

SCRIPTS BASH

GNU nano 2.9.3

script.sh

```
#!/bin/bash
mkdir ./dossier1
cd ./dossier1
touch fichier1
touch fichier2
```

```
chmod 700 script.sh
bash script.sh
```

EASY

Exercice

Créer un script bash qui crée un dossier test_bash, et qui crée trois fichiers à l'intérieur 1.txt 2.txt 3.txt puis qui copie le dossier test_bash vers un autre dossier test_bash_2

Créer une commande

un alias permet de créer des commandes personnalisées

```
alias [nom_alias]=[cmd]
```

Exercice : créer la commande
cree_deux_dossiers qui lance le script de
l'exercice précédent

Les variables



```
#!/bin/bash
var1="one"
var2="piece"
echo $var1 $var2
```

passage d'arguments

```
sssit@JavaPoint: ~
#!/bin/bash
#
echo this script name is $0
#
echo The first argument is $1
echo The second argument is $2
echo And third argument is $3
#
echo \$ \$\$ PID of the script
echo \# \$# Total number of arguments
"script.sh" 14 lines, 264 characters
```

\$1 - \$9 : arguments de la commande

./run_script.sh arg1 arg2 arg3

EASY

Exercice

Créer un script qui prend un chemin en premier argument et qui renvoie le nombre de mots cumulé de tous les fichiers qui composent le dossier

Opérations

```
#!/bin/bash  
  
var=$((3+9))  
echo $var
```

OPÉRATEUR	USAGE
+	ajout
-	soustraction
*	multiplication
/	division
**	exponentiation
%	module

Exercice

Créer un script bash qui prend 3 arguments : un nombre de victoires, d'égalité et de défaites d'une équipe de foot et qui renvoie son score : une victoire compte pour trois points, une égalité compte pour 1 point, une défaite enlève 1 point

Enregistrer l'output d'une commande dans une variable

```
var=$(cmd)
```

MEDIUM

EXERCICE

CRÉER UN SCRIPT QUI PREND UN URL DE PAGE WIKIPÉDIA EN PREMIER PARAMÈTRE,
QUI PREND UN MOT EN SECOND PARAMÈTRE
ET RENVOIE EN MESSAGE LE NOMBRE DE FOIS QUE LE MOT EST APPARU DANS LA PAGE WEB

MEDIUM

EXERCICE

```
#!/bin/bash
content_html=$(curl $1 | grep -o $2)
number_words=$(echo $content_html | wc -w)
echo "Le nombre de mot $2 que j'ai trouvé dans le code de la page est : $number_words"
```

CONDITIONS BASH

OPÉRATION	SYNTAXE	EXPLICATION
Égalité	num1 -eq num2	est num1 égal à num2
Supérieur à égal à	num1 -ge num2	est num1 supérieur à égal à num2
Plus grand que	num1 -gt num2	est num1 supérieur à num2
Moins qu'égal à	num1 -le num2	est num1 inférieur à égal à num2
Moins que	num1 -lt num2	est num1 inférieur à num2
Pas égal à	num1 -ne num2	est num1 différent de num2

```
read x
read y

if [ $x -gt $y ]
then
echo X is greater than Y
elif [ $x -lt $y ]
then
echo X is less than Y
elif [ $x -eq $y ]
then
echo X is equal to Y
fi
```

EASY

CONDITIONS BASH

CRÉER UN SCRIPT QUI PREND DEUX NOMBRES EN ARGUMENT, ET QUI RENVOIE LE MESSAGE "LESS_THAN_100" SI LA SOMME DES DEUX NOMBRES EST INFÉRIEURE À 100, "MORE_OR_EQUAL_THAN_100" SINON

CONDITIONS BASH

Créer un script qui prend deux arguments : le type du pokémon 1 et le type du pokémon 2 et qui renvoie en message l'efficacité de l'attaque

(3 types possibles : feu, eau, plante)

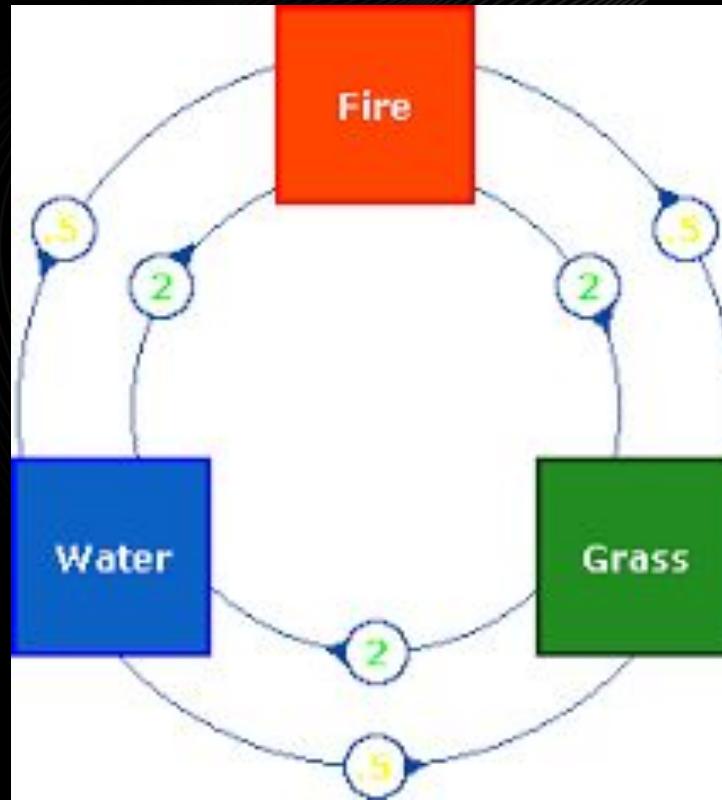
(Efficacité :

- Super efficace !
- Ce n'est pas très efficace !
- C'est efficace !

)



CONDITIONS BASH - EFFICACITE



SCRIPTS BASH - BOUCLES

Loops

Basic for loop

```
for i in /etc/rc.*; do  
    echo "$i"  
done
```

C-like for loop

```
for ((i = 0 ; i < 100 ; i++)); do  
    echo "$i"  
done
```

Ranges

```
for i in {1..5}; do  
    echo "Welcome $i"  
done
```

Reading lines

```
while read -r line; do  
    echo "$line"  
done <file.txt
```

Forever

```
while true; do  
    ...  
done
```

With step size

```
for i in {5..50..5}; do  
    echo "Welcome $i"  
done
```

BOUCLES EN BASH

```
#!/bin/bash
for i in {1..5}; do
    touch "${i}.txt"
done
```

SCRIPTS BASH - BOUCLES

```
#!/bin/bash
for i in {1..5}; do
    touch "${i}.txt"
done
```

RUN.SH

```
(base) eol@opFenice:~/Documents/travail/regexp/test$ ls
run.sh
(base) eol@opFenice:~/Documents/travail/regexp/test$ ./run.sh
(base) eol@opFenice:~/Documents/travail/regexp/test$ ls
1.txt 2.txt 3.txt 4.txt 5.txt run.sh
(base) eol@opFenice:~/Documents/travail/regexp/test$ █
```

SCRIPTS BASH - BOUCLES

HARDCORE



SCRIPTS BASH - BOUCLES

Mise en situation :

- 50% de la classe va se faire aspirer l'âme
- Pour déterminer qui vit et qui meurt :
- Vous avez 2 dés 20, vous pouvez juste décider un nombre de lancers
- A chaque lancer des deux dés, on additionne la somme des deux nombres et on ajoute cette valeur à un score total
- Les 50% de la classe avec les plus hauts score vivent
- Si un double apparaît, le score **FINAL** devient 0

DILEMME DU PRISONNIER

Se mettre par groupe de deux

- 50% de la classe va se faire emprisonner à vie
- Entre votre partenaire de cellule,
il y a une machine
- Il y a deux boutons sur cette machine
des deux côtés : “COOP” et “STEAL”
- COOP & COOP : 2 pieces chacun
- COOP & TRAHIR : 0 pieces 5 pieces
- TRAHIR & TRAHIR : 0 piece 0 piece

Il y a 6 ROUNDS DE JEU, afficher le score final

A la fin de la partie, les 50% de la classe avec le moins de pièces seront **enfermés à vie** et les autres gagneront leur **liberté**



SPEED TYPING TEST

HARDCORE

Créer un script type_fast.sh qui génère des mots aléatoires un par un et les affiche à l'utilisateur (utiliser le dico.txt)

L'utilisateur doit réécrire les mots générés le plus rapidement possible lettre par lettre. Tant que 20 mots n'ont pas été écrits correctement, on continue.

En utilisant la commande time, on affichera le temps d'exécution du script.

Qui sera le plus rapide?

http://www.3zsoftware.com/listes/liste_francais.zip



Planifier des tâches avec cron

- Lancer des scripts de façon régulière
- Choix de l'intervalle et de la date
- usermod -aG crontab (username)
- su username

```
fuokov2@opFenice:~$ crontab -e
```

Planifier des tâches avec cron

```
GNU nano 6.2                               /tmp/crontab.r5p8dW/crontab

# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
*
* * * * * mkdir /home/fuoko/test.cron
```

PLANIFIER DES TÂCHES AVEC CRON

crontab guru

The quick and simple editor for cron schedule expressions by [Cronitor](#)

“At every minute.”

[next](#) at 2022-12-12 11:08:00

[random](#)



minute	hour	day (month)	month	day (week)
--------	------	----------------	-------	---------------

*

any value

,

value list

'

separator

-

range of values

/

step values

@yearly (non-standard)

@annually (non-standard)

@monthly (non-standard)

@weekly (non-standard)

@daily (non-standard)

@hourly (non-standard)

@reboot (non-standard)

Planifier des tâches avec cron

- Une fois par mois
- Une fois tous les deux jours à 10 heures
- Le 6eme, 13eme et 24eme jour du mois à midi
- Toutes les trente minutes

Planifier des tâches avec cron

Créer une tâche cron mensuelle à 10 heures du matin qui fait une copie du contenu du dossier Documents/scripting vers un nouveau dossier “backup” dans le dossier personnel de l’utilisateur



Planifier des tâches avec cron

Créer une tâche cron qui lance un script bash toutes les minutes.



Le script est le suivant :

- Si le fichier `les_minutes_passent.txt` n'existe pas dans le répertoire personnel, il est créé avec comme contenu le chiffre 0.
- Sinon, ce fichier est remplacé en incrémentant le nombre présent dans l'ancien fichier de 1.

Variab les d'environnement

Pour afficher les variables d'env :

```
printenv
```

Pour créer une variable d'env :

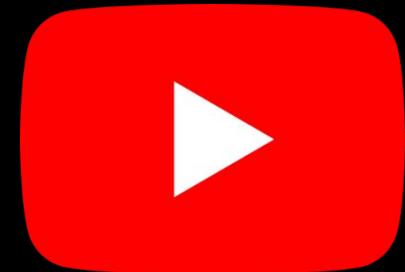
```
export env_var=10
```

```
echo $env_var
```

On peut aussi Modifier le fichier `~/.bashrc`

EXERCICE

CHERCHE MUSIQUE !



YOUTUBE TO MP3

Créer un script bash qui prend un nom de musique en paramètre et qui télécharge le fichier mp3 de la musique choisie

Indice : utiliser le package yt-dlp
pip install yt-dlp

usage : get_music “smells like teen spirit”
output : télécharge un mp3 de la musique

INTERAGIR AVEC UNE API



ChatGPT

Interagir avec une API #1 : GÉNÉRER UNE IMAGE

- Générer une clé API sur le site de chatgpt
- Sur l'api de génération d'images, tester la requête curl sur le terminal
- Mettre la requête curl dans un script bash et faire en sorte de pouvoir passer en paramètre du script la requête (le nom de l'image à générer)

Interagir avec une API #1 : GÉNÉRER UNE IMAGE

- Avec grep, extraire de la réponse du curl l'url vers l'image générée, et envoyer l'url à wget pour la sauvegarder dans un dossier images

Usage :

```
./generate_image.sh <texte de l'image>
```

Output :

enregistre l'image dans un dossier images

Interagir avec une API #2 : GÉNÉRER DU TEXTE

Utiliser cette fois-ci l'IA conversationnelle et refaire un script similaire

usage :

```
./generate_text.sh <requete>
```

output :

la réponse de l'IA sous forme de texte

Créer ensuite un alias “dischatgpt” et l'ajouter dans le .bashrc

Interagir avec une API #3 : CA PARLE !

Faire dialoguer deux ia conversationnelles à tour de rôle en prenant la réponse d'une ia et en l'envoyant à l'autre et ainsi de suite (sleep 10 entre chaque requête)

usage :

```
./discutez.sh <message_initial>
```

output :

un dialogue entre deux IA

Interagir avec une API #4 : CA IMAGINE ...

Demander à l'ia conversationnelle d'imaginer et de décrire une situation ou une image et donner sa réponse en entrée de l'ia qui génère les images

usage :

```
./imagine.sh <requete imagination>
```

output :

une image qui est le résultat de ce qu'a décrit l'ia conversationnelle

INTERAGIR AVEC UNE API # CHERCHE RECETTE



Faire un script qui prend une recette en paramètre et qui renvoie tous les ingrédients un par un.

Usage :

```
get_recette "boeuf bourguignon"
```

Output :

“ Voici les ingrédients du boeuf bourguignon :

- 700G de paleron de boeuf
- 100G de Farine
- 1/2L de vin rouge...

MAIL - APT INSTALL SSMTDP
CONFIG /ETC/SSMTP/SSMTP.CONF
SENDMAIL LEO.SUP@HOTMAIL.FR "OK T'AS VU CA
MARCHE AUSSI"
TELNET SMTP.GMAIL.COM 587
SUDO APT INSTALL POSTFIX

```
##  
# Config file for sSMTP sendmail  
#  
# The person who gets all mail for userids < 1000  
# Make this empty to disable rewriting.  
SERVER=fuoko.fr@gmail.com  
mailhub=smtp.gmail.com:587  
AuthUser=fuoko.fr@gmail.com  
authPass=mohmngpwipzibbrs  
UseSTARTTLS=YES
```