Systèmes d'exploitation 420-W12-SF

Utilisation des scripts Bash

Jean-Pierre Duchesneau, Automne 2021

Les cours

1. Intro aux infrastructures informatiques et les composantes internes du PC
2. Les composantes internes du PC et réseau
3. Système d'Exploitation
4. Virtualisation de clients et de serveurs
5. Disque dur, partition et système de fichier
6. La ligne de commandes (Shell) et les scripts
    1. CMD
    2. Linux
    3. Bash
    4. Les scripts
7. **Git, le contrôle de version**
8. WAMP

Un script

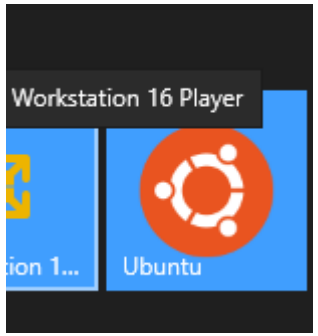
- Contient une série de commandes.
- Ces commandes sont exécutés par un interpréteur les unes après les autres.
- Tout ce que vous pouvez taper en ligne de commande peut être inclus dans un script.
- Le scripting est la méthode idéale pour l'automastisation de tâches, notamment dans la mouvance DevOps

Langage de scripts les plus utilisés

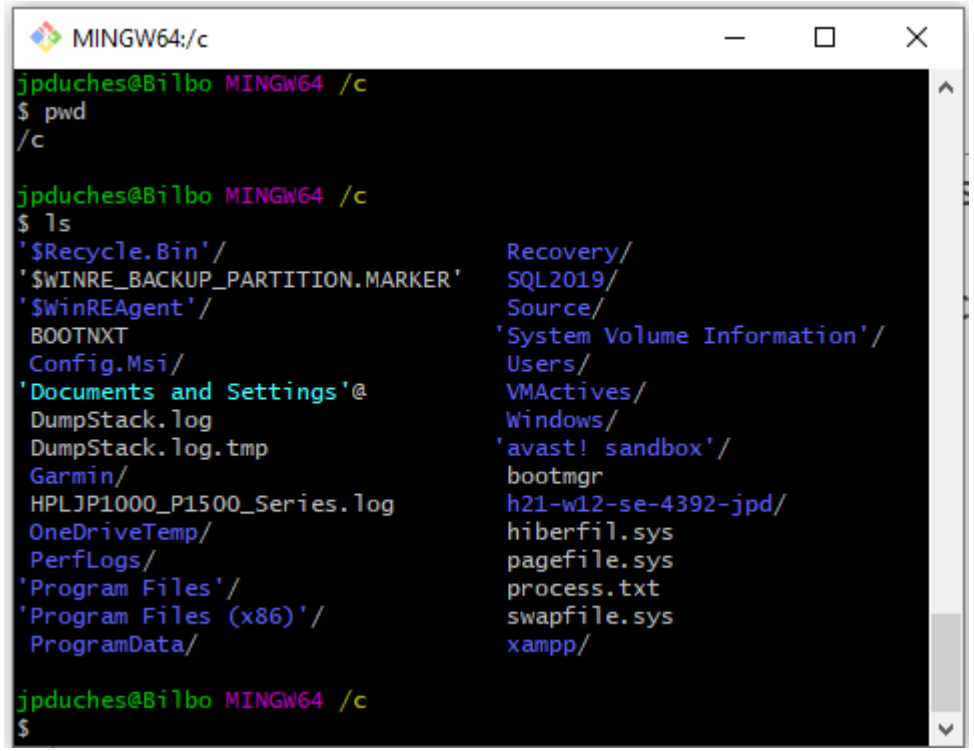
- Bash
- Power Shell
- Python

Bash

- **Bash** (acronyme de ***Bourne-Again shell***) est un interpréteur en ligne de commande de type script.
- C'est le shell Unix du projet GNU.
- Bash est un logiciel libre publié sous licence publique générale GNU.



Ubuntu sous Windows



Bash sous Windows

L’aide sur Bash

La documentation de Bash est disponible en ligne, comme celle de la plupart des logiciels GNU.
https://www.gnu.org/software/bash/

Vous pouvez également trouver des informations sur Bash en executant

- **info bash** ou
- **man bash**, ou en consultant
- /usr/share/doc/bash/, /usr/local/share/doc/bash/,
- ou des répertoires similaires sur votre système.
- Un résumé est disponible en exécutant bash --help.

Rendre le script exécutable

- Modifier les droits : **chmod a+x script.sh**
- Le chemin d'exécution :

Chemin absolu :
/home/jpduches/script.sh

Chemin relatif (si je suis dans le même répertoire que le script) :
./script.sh

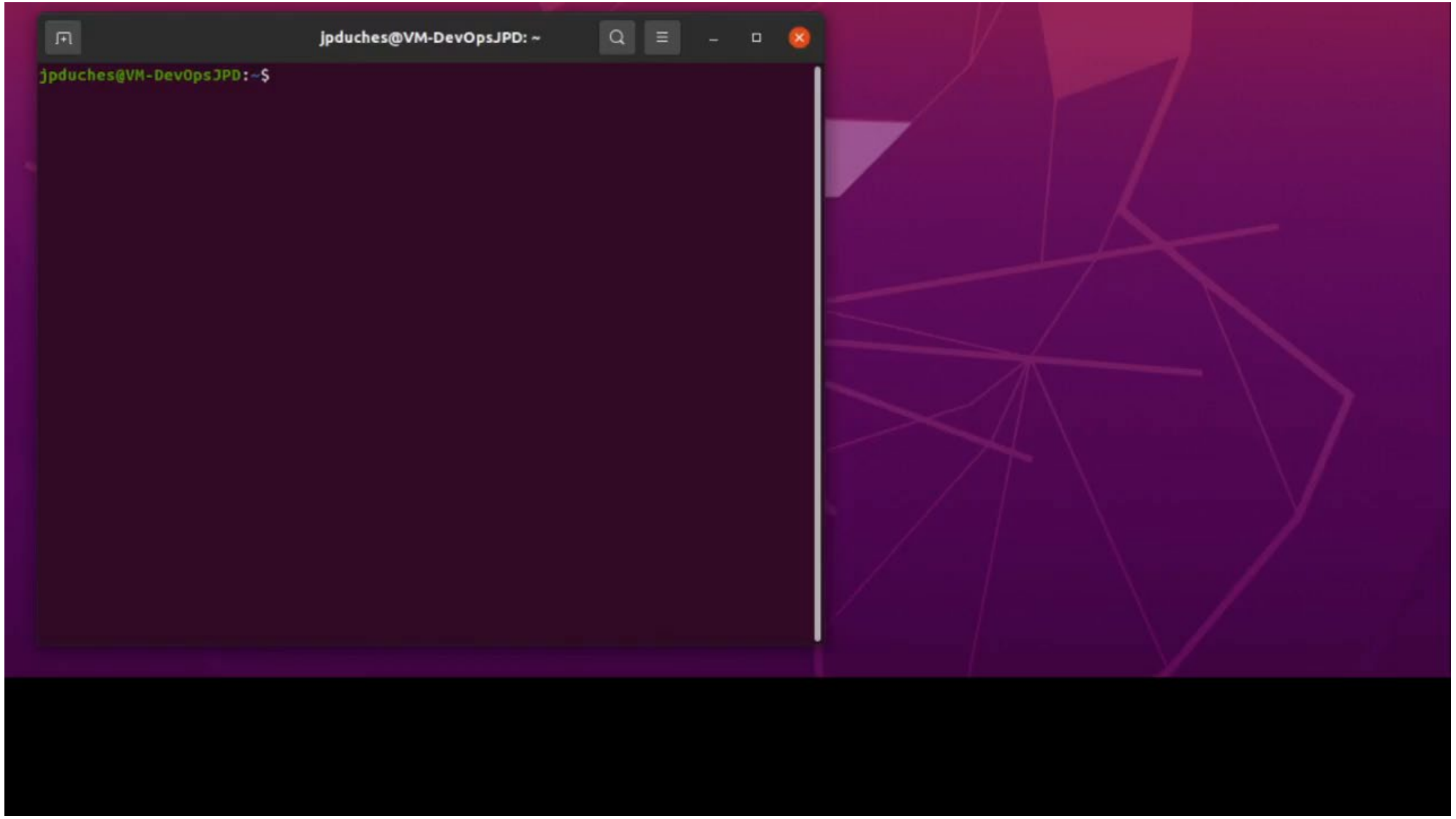
- Le shebang au début du script bash :

```
#!/bin/bash
```

- D'autres shebang

```
#!/bin/sh
#!/bin/csh
#!/bin/zsh
…
```

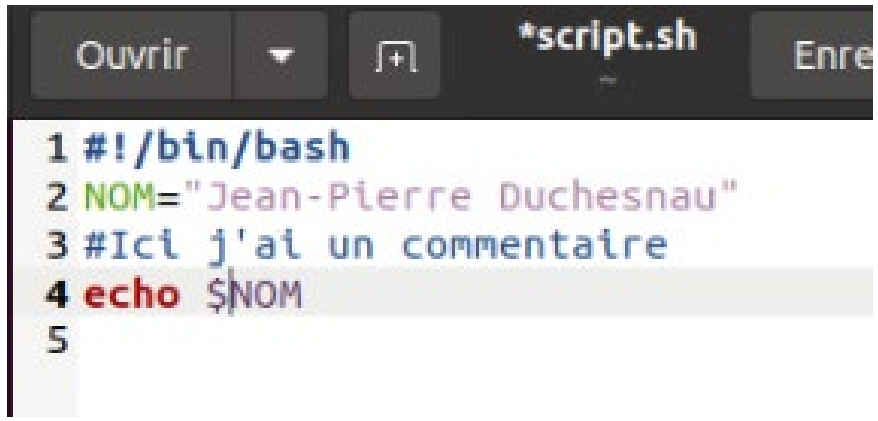
Exemple avec un autre interpréteur :

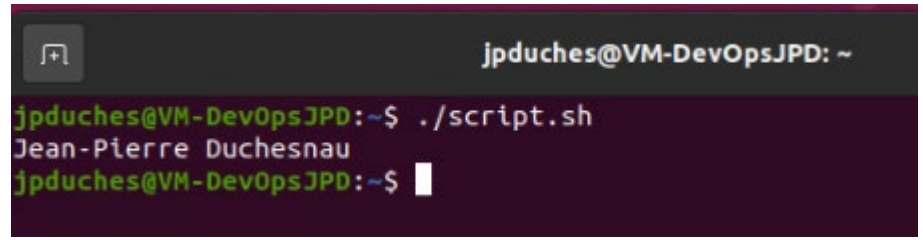


Les variables et commentaires

- Les variables sont sensibles à la casse et par convention on le mot toujours en majuscules.
- Attention à ne pas mettre d'espace entre les variables, le signe = et les ".

NOM_DE_LA_VARIABLE="valeur"



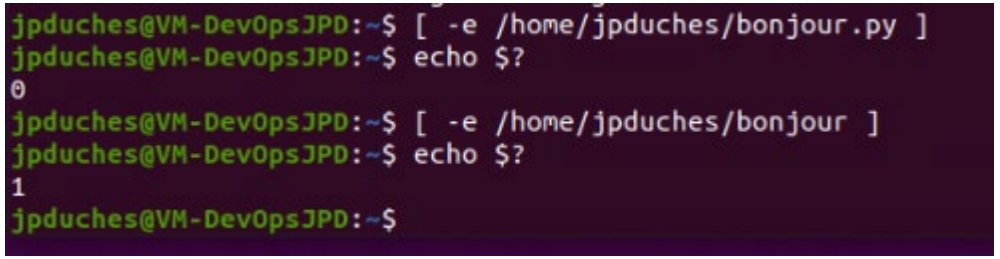


Les tests

Exemple de tests :

Vérifier si le fichier /home/jpduches/bonjour existe. Renvoi 0 (true) 1 (false)

```
[ -e /home/jpduches/bonjour ]
$echo $?
```



Opérateur principaux :

- -e : (True) si le fichier existe
- -d : (True) s'il s'agit d'un dossier
- -r : (True) si le fichier est disponible en lecture pour l'utilisateur
- -s : (True) si le fichier existe et n'est pas vide
- -w : (True) si le fichier est disponible en écriture pour l'utilisateur
- -x : (True) si le fichier est disponible en exécution pour l'utilisateur

```
$help test
```

Variable de positionnement

- Stockent le contenu
- Il en existe 10 : $0 jusqu'à 9
- Le script lui-même est stocké dans la variable $0
- Le premier paramètre est stocké dans la variable $1
- Le second paramètre est stocké dans la variable $2
- Si plus de 9 arguments on utilise la commande shift (voir page suivante)

```
#!/bin/bash

echo  "Le  premier argument a pour valeur $1"
echo "Le second argument a pour valeur  $2"


echo "les arguments ont pour valeur : $@"
echo "Le  nombre d'argument est de $#"

echo "Les arguments ont pour valeurs $*"
```

```
jpduches@VM-DevOpsJPD:~$ ./position.sh Jean-Pierre Duchesneau
Le  premier argument a pour valeur Jean-Pierre
Le second argument a pour valeur  Duchesneau
les arguments ont pour valeur : Jean-Pierre Duchesneau
Le  nombre d'argument est de 2
Les arguments ont pour valeurs Jean-Pierre Duchesneau
jpduches@VM-DevOpsJPD:~$
```

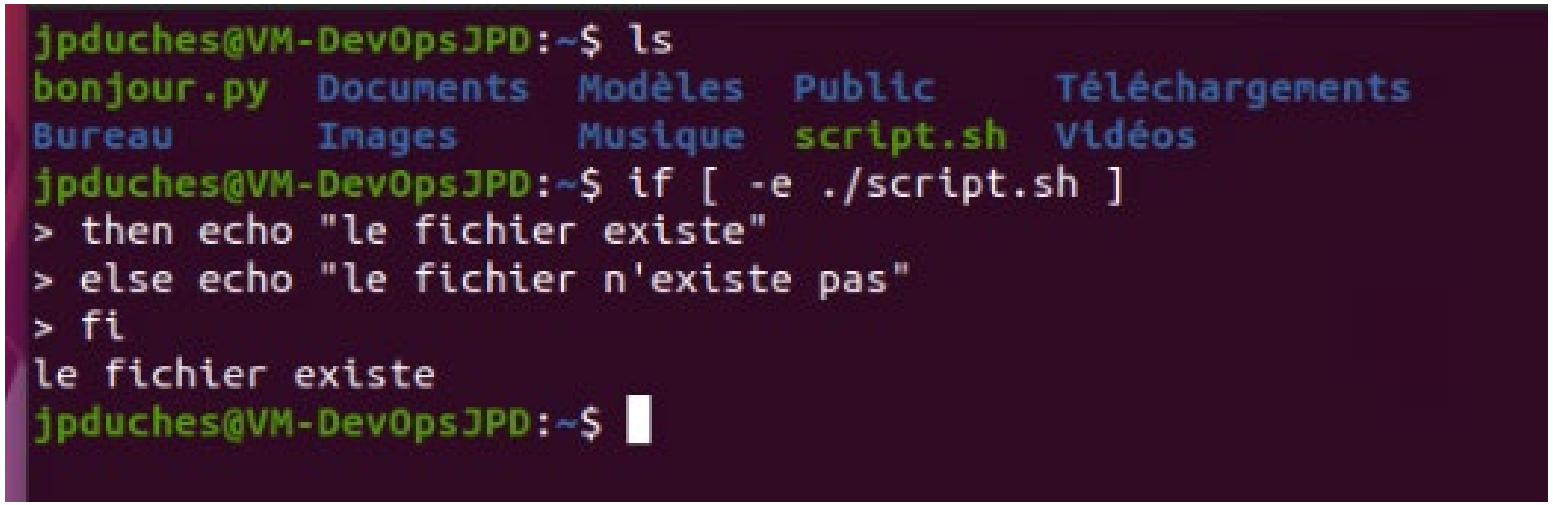
$# : récupère le nombre de paramètres (à partir du $1)
$* : récupère la liste des paramètres

Les conditions : if, elif,else

```
if [condition-est-vrai]
then
        command
else
        command
fi
```

Si la condition est vraie, les commandes situées après le then sont exécutées.

Si la condition est fausse, les commandes situées après le else sont exécutées.



```bash
#!/bin/bash

CHIFFRE1='16'
CHIFFRE2='15'

if [ $CHIFFRE1 -lt $CHIFFRE2 ]
then
echo  "$CHIFFRE! est plus petit que  $CHIFFRE2"

elif  [ $CHIFFRE1 -gt $CHIFFRE2 ]
then
echo "$CHIFFRE1 est plus grand  que $CHIFFRE2"

else
echo "$CHIFFRE 1 est égale  à $CHIFFRE2"
fi
```

```
jpduches@VM-DevOpsJPD:~$ ./script.sh
16 est plus grand  que 15
jpduches@VM-DevOpsJPD:~$
```

Boucles :

for

do

done

```
#!/bin/bash

CHIFFRES="10 11 12 13 14"

for   CHIFFRE   in   $CHIFFRES
do
echo "CHIFFRE   :   $CHIFFRE"
done
```

```
jpduches@VM-DevOpsJPD:~$ ./boucle.sh
CHIFFRE   :   10
CHIFFRE   :   11
CHIFFRE   :   12
CHIFFRE   :   13
CHIFFRE   :   14
jpduches@VM-DevOpsJPD:~$
```

While [la condition-est-vraie]

Do

- Command

done

```
#!/bin/bash

while [ -z $PRENOM ]
do
read -p "Quel est votre prnénom ? "  PRENOM
done

echo "Votre prénom est $PRENOM"
```

```
jpduches@VM-DevOpsJPD:~$ ./while.sh
Quel est votre prnénom ?
Quel est votre prnénom ?
Quel est votre prnénom ? Jean-Pierre
Votre prénom est Jean-Pierre
jpduches@VM-DevOpsJPD:~$
```

Fonctions

```
#!/bin/bash

function internet () {
ping -c 1  8.8.8.8

if  [ $? -eq 0 ]
then
        echo "La connectivité  vers internet est établie"
else
        echo " Pas de connectivité vers internet"
fi
}

internet
```

```
jpduches@VM-DevOpsJPD:~$ ./fonction.sh
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 octets de 8.8.8.8 : icmp_seq=1 ttl=116 temps=5.56 ms

--- statistiques ping 8.8.8.8 ---
1 paquets transmis, 1 reçus, 0 % paquets perdus, temps 0 ms
rtt min/avg/max/mdev = 5.556/5.556/5.556/0.000 ms
La connectivité  vers internet est établie
```