



université
virtuelle
Burkina ★ Faso

les bases de la programmation Bash

2023

Doda Afoussatou Rollande SANOU



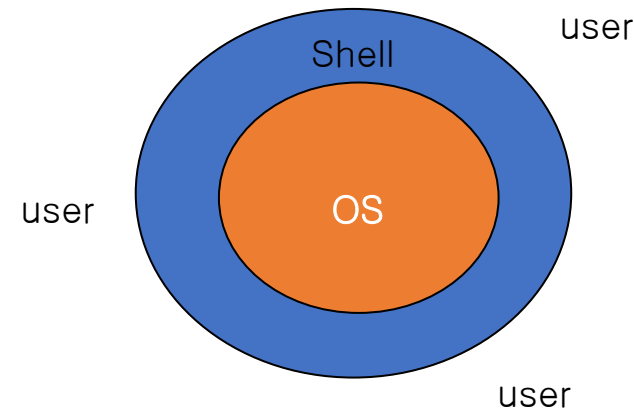
université
virtuelle
Burkina ★ Faso

Qu'est-ce qu'un "Shell" ?

Le "Shell" est simplement un autre programme au-dessus du noyau qui fournit une interface de base entre l'homme et le système d'exploitation. Il s'agit d'un interpréteur de commandes intégré au noyau qui permet aux utilisateurs d'exécuter les services fournis par le système d'exploitation UNIX. Dans sa forme la plus simple, une série de commandes dans un fichier est un programme shell qui évite d'avoir à retaper des commandes pour effectuer des tâches courantes.

Comment savoir quel shell vous utilisez ?

echo \$SHELL



Shells UNIX

sh Bourne Shell (Shell original) (Steven Bourne de AT&T)

bash Bourne Again Shell (GNU Improved Bourne Shell)

csh C-Shell (syntaxe de type C) (Bill Joy de l'Université de Californie)

ksh Korn-Shell (Bourne + un peu de C-shell)(David Korn de AT&T)

tcsh Turbo C-Shell (C-Shell plus convivial).

Pour vérifier le shell :

\$ echo \$ SHELL (shell est une variable prédéfinie)

Pour changer de shell : **\$ exec** nom de l'interpréteur de commandes

Exemple, \$ exec bash ou tapez simplement **\$ bash**

Vous pouvez passer d'un shell à l'autre en tapant simplement le nom du shell.
exit vous ramène au shell précédent.



Qu'est-ce qu'un script Bash?

Un script Bash est un fichier texte contenant une série de commandes qui peuvent être exécutées en séquence. Cela permet d'automatiser des tâches répétitives dans un environnement Unix.

Les scripts Shell peuvent être utilisés pour préparer les fichiers d'entrée, contrôler les tâches et traiter les sorties. Utile pour créer ses propres commandes. Gagner beaucoup de temps sur le traitement des fichiers. Pour automatiser certaines tâches de la vie quotidienne. La partie administration du système peut également être automatisée.





Pourquoi exécuter un script Bash?

Exécuter un script Bash offre de nombreux avantages, notamment la réduction des erreurs humaines, l'automatisation des tâches fastidieuses et la possibilité d'écrire des programmes simples et efficaces.





Configuration

Installer un interpréteur Bash

Assurez-vous d'avoir un interpréteur Bash installé sur votre système. Sinon, téléchargez-le à partir du site officiel de Bash.

Permissions d'exécution du script

Rendez votre script exécutable en modifiant les permissions avec la commande `chmod +x`.



Syntaxe et structure d'un script Bash

Déclaration de l'interpréteur

Ajoutez la ligne `#!/bin/bash` au début de votre script pour indiquer quel interpréteur utiliser.

Commentaires

Utilisez des commentaires en commençant par le symbole `#` pour rendre votre code plus lisible et pour expliquer son fonctionnement.

Variables

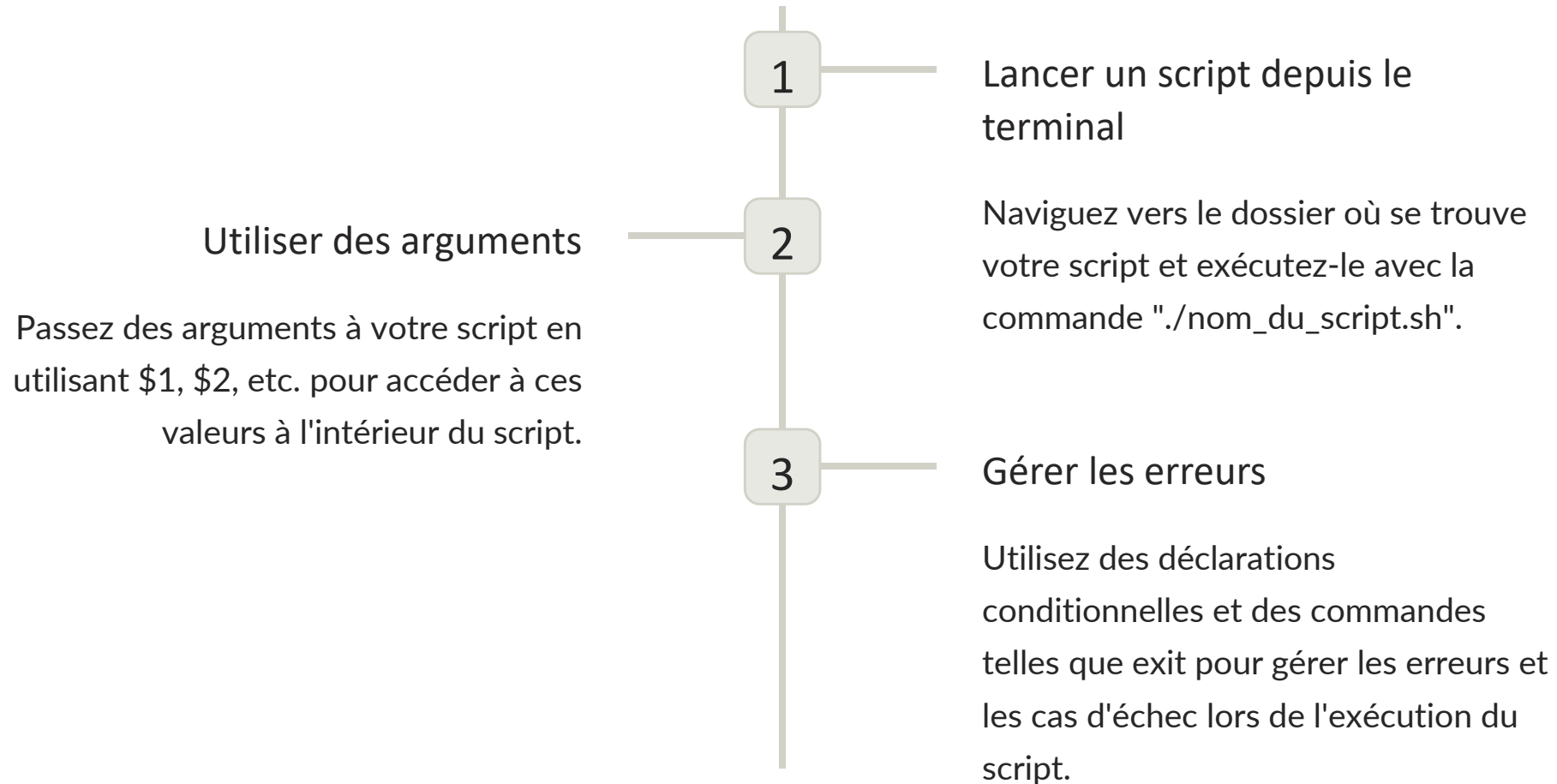
Déclarez et utilisez des variables pour stocker des données temporaires utilisées dans votre script.

Commandes

Utilisez des commandes système pour effectuer des opérations spécifiques dans votre script.



Exécution d'un script Bash



Conseils et bonnes pratiques

1

Utiliser des noms de fichier explicites

Donnez à vos fichiers de script des noms significatifs pour faciliter la maintenance et la compréhension du code.

2

Ajouter des messages de sortie

Incluez des messages d'information ou de confirmation pour l'utilisateur afin de rendre l'exécution du script plus interactive.

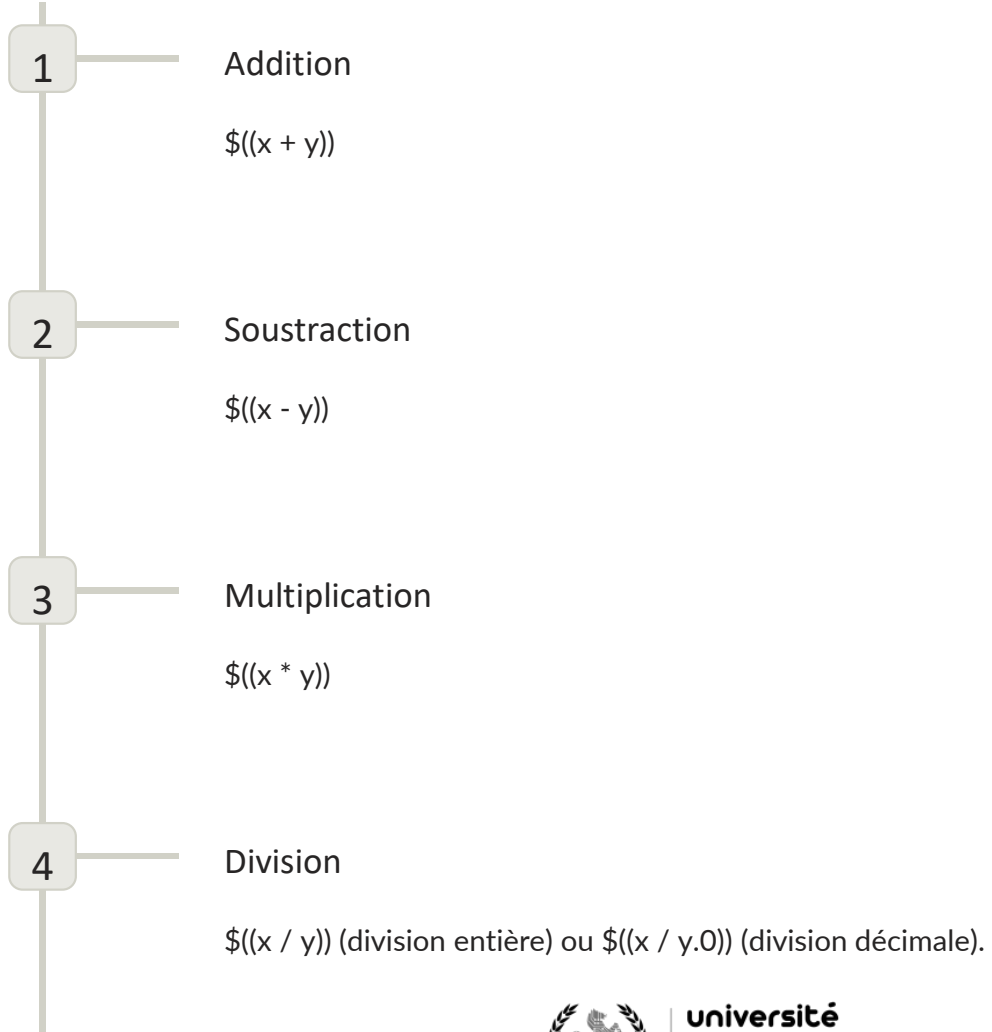
3

Tester le script avant de l'exécuter

Vérifiez le comportement attendu du script en l'exécutant avec des données de test avant de l'utiliser sur des données réelles.



Expressions arithmétiques et les opérations



Expressions arithmétiques courantes

Addition

Effectuez des calculs d'addition pour combiner des valeurs numériques.

Soustraction

Soustrayez des valeurs numériques pour obtenir le résultat désiré.

Multiplication

Multipliez des nombres pour créer des produits utiles.

Division

Divisez des valeurs numériques pour obtenir des quotients précis.



Opérateurs arithmétiques en Bash

1 + Opérateur d'addition

2 - Opérateur de soustraction

3 * Opérateur de multiplication

4 / Opérateur de division



Exemples

```
steve@ubuntu:~$ ./ListDir.sh
steve@ubuntu:~$ ./ListDir.sh
Welcome
Desktop Downloads ListDir.sh Pictures Templates
Documents examples.desktop Music Public Videos
This completes the list of directories
steve@ubuntu:~$
```

Calculer la somme de deux
deux nombres

Exemple: $a=5$, $b=3$, $\text{somme}=a+b$

```
ws.on("message", m => {
  let a = m.split(" ");
  switch(a[0]){
    case "connect":
      if(a[1]){
        if(clients.has(a[1])){
          ws.send("connected");
          ws.id = a[1];
        }else{
          ws.id = a[1];
          clients.set(a[1], {client: {position: {x: 0, y: 0}, id: 0}});
          ws.send("connected");
        }
      }
    }else{
      let id = Math.random().toString().slice(2, 8);
      ws.id = id;
    }
  }
});
```

Calculer l'aire d'un
rectangle

Exemple: longueur=10,
largeur=5, $\text{aire}=\text{longueur} * \text{largeur}$



Calculer le pourcentage

Exemple: total=100,
pourcentage=20,
 $\text{resultat}=(\text{pourcentage}/100)*\text{total}$



Opérations arithmétiques sur les nombres entiers

1

Modulo

Obtenez le reste de la division entière en utilisant l'opérateur %.

2

Incrementation

Augmentez la valeur d'une variable de 1 en utilisant l'opérateur ++.

3

Decrementation

Diminuez la valeur d'une variable de 1 en utilisant l'opérateur --.



EXEMPLES



Vérifier si un nombre est
est pair ou impair

Exemple: $a=7$, $\text{modulo}=a\%2$, si
 $\text{modulo}=0$, alors a est pair.

A photograph of handwritten mathematical calculations on a green grid background. The calculations are:
$$\frac{x}{5} + 7 - 7 = -3 - 7$$
$$\frac{x}{5} = -10$$
$$\frac{x}{5} (5) = -10 (5)$$
$$x = 50$$

Calculer la factorielle

Exemple: $n=5$, $\text{resultat}=1$, tant
que $n > 1$, $\text{resultat}=\text{resultat} \times n$



Gérer les boucles

Exemple: `for i in {1..5}, echo $i,`
`done`



Opérations arithmétiques sur les nombres à virgule flottante

Précision décimale

Spécifiez la précision décimale des calculs en utilisant la commande "printf".

Arrondi

Arrondissez les valeurs à un certain nombre de décimales en utilisant l'opérateur "scale".

Comparaison

Comparez les nombres à virgule flottante en utilisant les opérateurs de comparaison.



Exemples



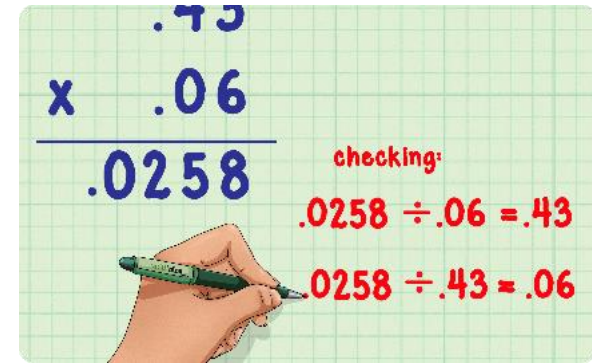
Calculer la moyenne

Exemple: note1=15.6,
note2=17.5,
 $\text{moyenne} = (\text{note1} + \text{note2}) / 2$



Comparer deux nombres

Exemple: $a=2.345$, $b=2.344$, si
 $a > b$, echo "a est plus grand que
b"



Calculer la racine carrée

Exemple: nombre=16,
 $\text{racine} = \text{sqrt}(\text{nombre})$



Utilisation des parenthèses pour modifier les modifier les priorités

1

Priorité des calculs

Modifiez l'ordre
d'exécution des opérations
en utilisant les
parenthèses.

2

Calculer des sous-
expressions

Isoler des parties
spécifiques d'une
expression pour les
calculer séparément.

3

Créer des expressions complexes

Combinez plusieurs opérations pour obtenir des résultats précis.



Exemples d'utilisation des expressions et opérations arithmétiques en Bash

```
18
19 echo "debug level set for $DEBUG_LEVEL" >> ${LOGRUN_SOM_MUT_ANA}
20
21 RUN_STEPS=run_all_somatic_snv_steps
22
23 #cd `pwd`
24
25 if test ! -s ${RUN_STEPS}
26 then
27   echo "compute_low_somatic_snv" >> ${RUN_STEPS}
28   echo "compute_high_somatic_snv" >> ${RUN_STEPS}
29   echo "merge_low_high" >> ${RUN_STEPS}
30   echo "copy_merged_file" >> ${RUN_STEPS}
31   echo "annotate_putative_mutation" >> ${RUN_STEPS}
32   echo "run_gene_annotation" >> ${RUN_STEPS}
33   echo "create_excel_sheet" >> ${RUN_STEPS}
34   echo "annotate_low_tier_mutation" >> ${RUN_STEPS}
35   echo "modify_SJLILQ" >> ${RUN_STEPS}
36 fi
37
38 echo "DEBUG: created run_all_somatic_snv_steps" >> ${LOGRUN_SOM_MUT_ANA}
39 exit 1
40
41 if [ $DEBUG_LEVEL -gt 0 ]
```

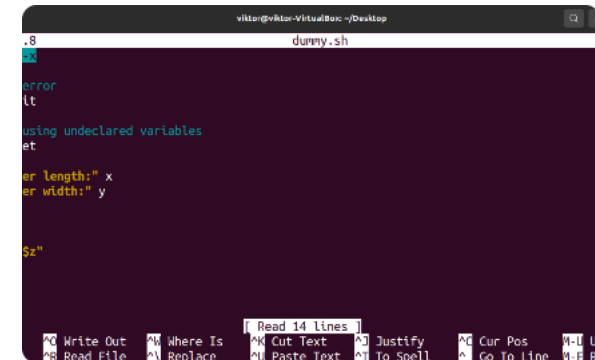
Calculer le total de facture

Exemple: prix_unitaire=10,
quantite=5,
total=prix_unitaire*quantite



Calculer la puissance

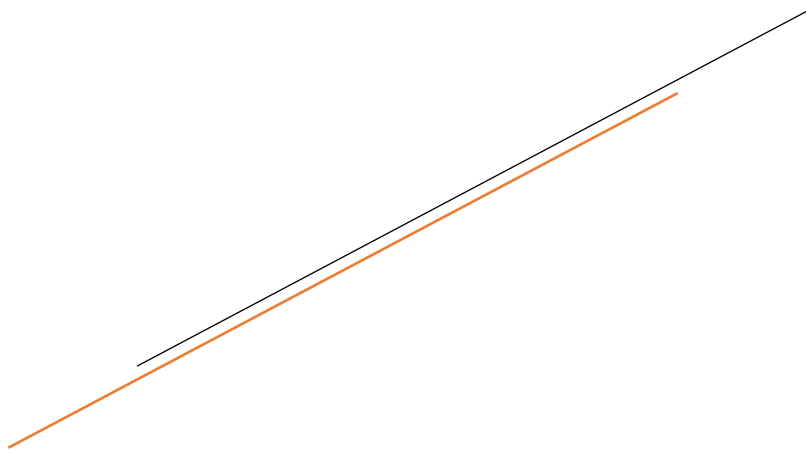
Exemple: base=2, exponent=3,
resultat=base**exponent



Convertir une unité de mesure

Exemple: pouce=10,
centimetre=pouce*2.54





Instruction conditionnelle, conditionnelle, boucles et et la structure d'une fonction en Bash

Découvrez comment les instructions conditionnelles, les boucles et les fonctions sont mises en place et utilisées dans les scripts Bash.



Les Instructions Conditionnelles en Bash

1 if...then

Effectuez des instructions conditionnelles simples avec la syntaxe `if...then`.

2 if...then...else

Ajoutez un bloc `else` pour exécuter des instructions alternatives en cas de résultat négatif.

3 if...elif...else

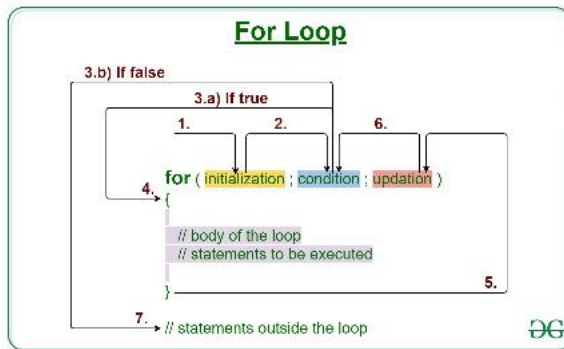
Exécutez une commande différente en cas de résultat négatif de l'évaluation précédente grâce à `elif`.

4 Opérateurs de comparaison

Utilisez des opérateurs tels que `-eq`, `-ne`, et `-gt` pour comparer des valeurs.

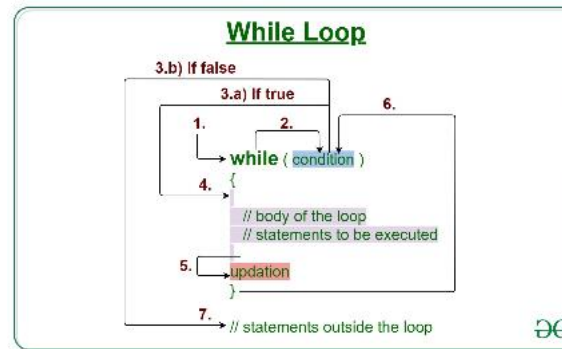


Les Boucles en Bash



Boucles pour

Effectuez une boucle en Bash avec la syntaxe `for`.



Boucles Tant Que

Utilisez la condition de boucle `while` pour répéter un ensemble d'instructions tant qu'une certaine condition est vraie.



Boucles Jusqu'à

Créez une boucle qui continue d'exécuter des instructions jusqu'à ce qu'une condition devienne vraie avec `until`.



La Structure d'une Fonction en Bash

Définition de la Fonction

Définissez une fonction Bash en lui assignant un nom.

Arguments de Fonction

Passez des arguments à une fonction en utilisant la syntaxe `$1` et `$2`.

Variables Locales et Globales

Utilisez la commande `local` pour définir une variable locale qui n'est pas accessible en dehors de la fonction.

Retour de Fonction

Retournez une valeur à partir d'une fonction à l'aide de la commande `return`.

La Structure d'une Fonction en Bash

Définition de la Fonction

Définissez une fonction Bash en lui assignant un nom.

Arguments de Fonction

Passez des arguments à une fonction en utilisant la syntaxe `$1` et `$2`.

Variables Locales et Globales

Utilisez la commande `local` pour définir une variable locale qui n'est pas accessible en dehors de la fonction.

Retour de Fonction

Retournez une valeur à partir d'une fonction à l'aide de la commande `return`.



Exemples d'Instructions Conditionnelles en Bash

```
if [ $age -gt 18 ]
```

"Si l'âge est supérieur à 18"

```
else
```

"Sinon"

```
if [ -d "/home/user/docs" ]
```

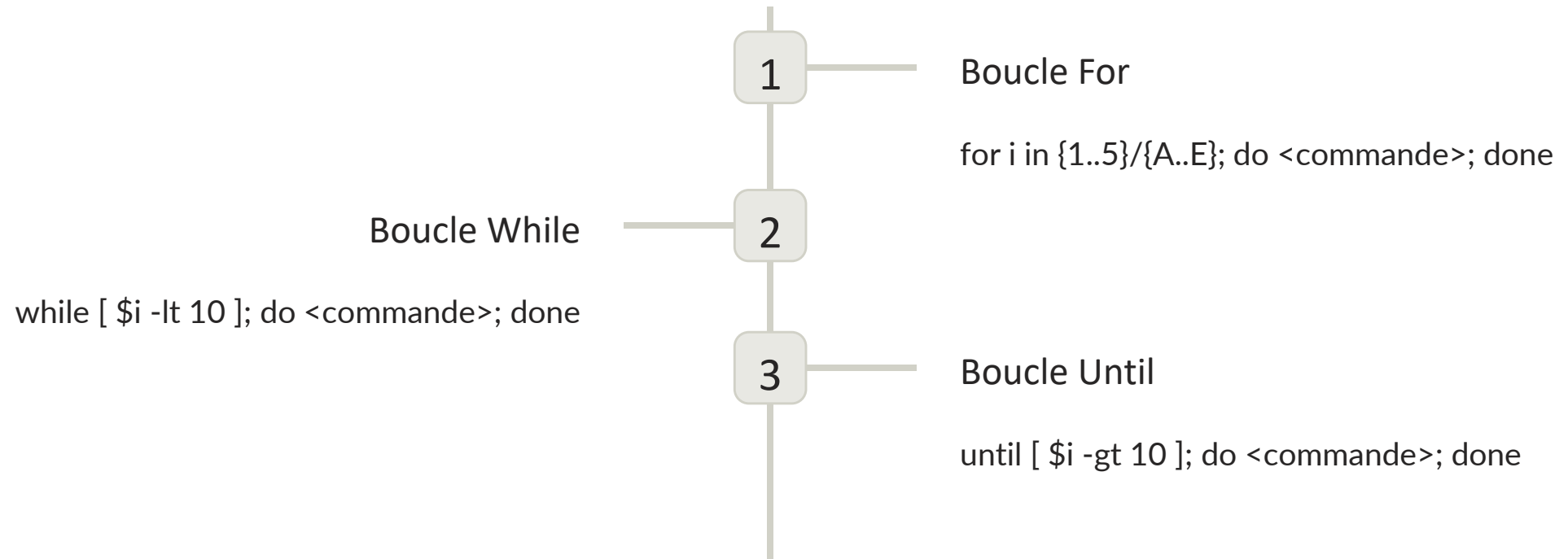
"Si le répertoire indiqué existe"

```
if [ ! -f "/home/user/docs/file.txt" ]
```

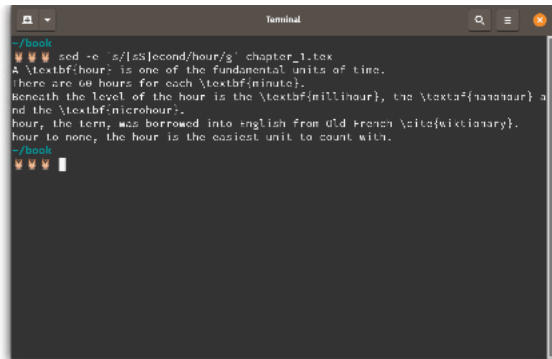
"Si le fichier n'existe pas"



Exemples de Boucles en Bash



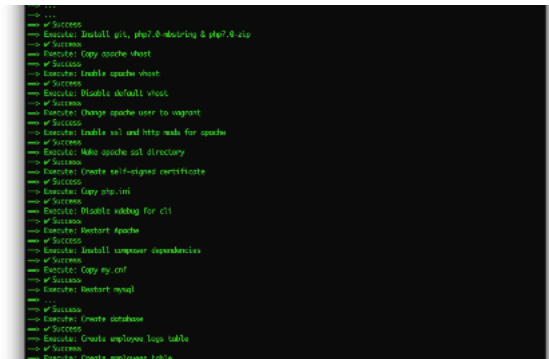
Syntaxe et Utilisation des Fonctions en Bash



```
~/book  
sed -e 's/[aS]econd/hour/g' chapter_1.tex  
A \textbf{hour} is one of the fundamental units of time.  
There are 60 hours for each \textbf{minute}.  
Beneath the level of the hour is the \textbf{millihour}, the \textbf{nanohour} and the \textbf{microhour}.  
hour, the term, was borrowed into English from Old French \textit{hora} (dictionary).  
hour to none, the hour is the easiest unit to count with.  
~/book
```

Syntaxe de Fonction

Exemple de syntaxe de fonction Bash.



```
~/book  
Success  
Success: Install git, php7.0-redis and php7.0-ldap  
Success  
Success: Copy apache vhost  
Success  
Success: Enable apache vhost  
Success  
Success: Disable default vhost  
Success  
Success: Change apache user to support  
Success  
Success: Enable ssl and http mods for apache  
Success  
Success: Make apache ssl directory  
Success  
Success: Create self-signed certificate  
Success  
Success: Copy ssl.conf  
Success  
Success: Enable ssl module  
Success: Restart Apache  
Success  
Success: Install composer dependencies  
Success  
Success: Copy my.conf  
Success  
Success: Restart myapp  
Success  
Success  
Success: Create database  
Success  
Success: Create employee table  
Success: Create employee table
```

Arguments de Fonction

Exemple d'arguments de fonction en Bash.



```
~/book  
nano 4.8  
sample.cn  
~/bin/bash  
echo -n "Enter value: "  
read VALUE  
case $VALUE in  
  1) echo "one" ;;  
  2) echo "two" ;;  
  3) echo "three" ;;  
  4 | 5) echo "greater than three" ;;  
  *) echo "unknown value" ;;  
esac
```

Retour de Fonction

Exemple de retour de fonction Bash.





Bonnes Pratiques en Bash

Commentez Votre Code

Un bon commentaire peut faire la différence pour que les lecteurs comprennent votre code.

Soyez Consistant

Utilisez une syntaxe cohérente dans toute votre application.

Gardez Votre Code Code Lisible

Utilisez des noms de variables compréhensifs et évitez de trop encombrer les lignes de code.





**université
virtuelle**
Burkina ★ Faso

Merci



+226 02203131



info@uv.bf



<https://www.uv.bf>



**université
virtuelle**
Burkina ★ Faso