



UNIX

Cours 7 : Programmation SHELL (1)

Filipe Vasconcelos¹
¹ESME, Lille, filipe.vasconcelo@esme.fr

- ① Rappel sur le SHELL**
- ② SHELL script**
- ③ Premier Script**
- ④ Les variables en bash**



AAV4

Produire, tester et vérifier le résultat des scripts SHELL (en bash) pour réaliser des tâches algorithmiques simples à complexes



① Rappel sur le SHELL

② SHELL script

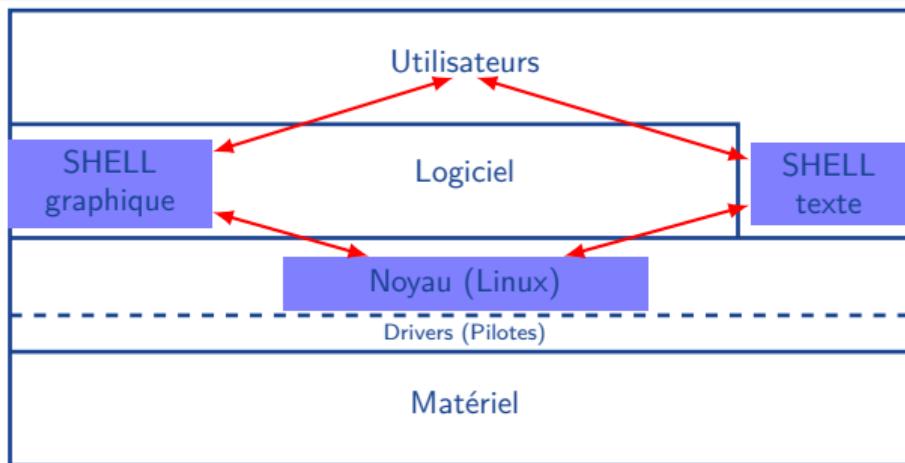
③ Premier Script

④ Les variables en bash



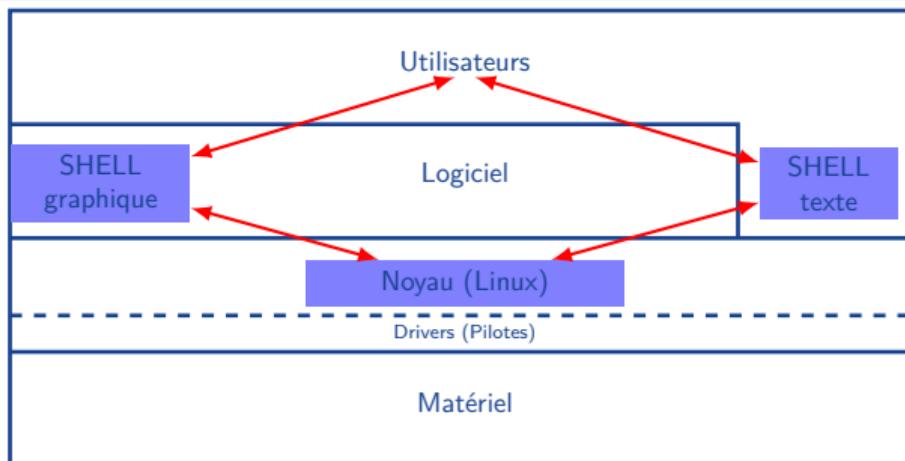
GUI : *Graphical User Interface*

- ❑ Il existe plusieurs environnements graphiques sous GNU/Linux :
 - ❑ GTK : Gnome, Unity, MATE, Xfce...
 - ❑ Qt : KDE, LXQt, ...
 - ❑ WSL (Windows) : Xming



CLI : *Command line interface*

- ❑ Il existe différentes versions d'interpréteur de commande :
 - ❑ bash : Bourne Again Shell



① Rappel sur le SHELL

② SHELL script

③ Premier Script

④ Les variables en bash



Le SHELL script

Définition

- Les instructions sont écrits dans un fichier appelé « shell script » ou encore simplement script.
- Séquences de commandes (batch) au contraire du mode interactif de l'invite de commande.

Usage

- Accès direct au système et à l'ensemble des commandes.
- Automatisation de tâches complexes et répétitives
- Création de nouvelles commandes (« un alias avancé »)
- Administration Système

Particularités

- Attention !!! Syntaxe stricte !
- Attention !!! Lisibilité !

- ① Rappel sur le SHELL
- ② SHELL script
- ③ Premier Script
- ④ Les variables en bash



Nom du script

- ❑ Par convention les shell script ont généralement pour extension .sh.
- ❑ Exemple : `mon_premier_script.sh`

La première ligne du script

- ❑ Un script commence par le nom de l'interpréteur du script.
- ❑ Les deux premiers caractères sont #! (« shabang »)
- ❑ Ensuite le chemin d'accès vers l'interpréteur. Exemple : `#!/bin/bash`
- ❑ Fun Fact : Vous pouvez commencer un script par d'autres interpréteurs.
 - ❑ awk : `#!/usr/bin/awk`
 - ❑ perl : `#!/usr/bin/perl`
 - ❑ python : `#!/usr/bin/python3`



Les commentaires

Les lignes commentées commencent par #.

```
#!/bin/bash
#Ceci est mon premier script bash
#Il permet d'afficher le message "Hello World!"
echo "Hello World!"
```

Exécution du script

- Donner les droits d'exécution au script avec la commande :
`chmod u+x votre_nom_script`
- Puis taper ./ devant le nom du script ou taper le chemin absolu de ce script pour l'exécuter.

- ① Rappel sur le SHELL
- ② SHELL script
- ③ Premier Script
- ④ Les variables en bash



Une syntaxe particulière

- ❑ Toute variable possède un nom et une valeur : `nom=valeur` Exemple : `maVariable="Bonjour"`
- ❑ Remarque !! : Il n'y a pas d'espace à gauche ou à droite du signe égal
- ❑ Attention : on peut utiliser des variables non initialisées qui sont alors égales à la chaîne vide

Initialisation de variable à partir du résultat d'une commande

- ❑ Avec des accents graves (`AltGr+7`). Exemple : `maVariable=`ls``
- ❑ Avec des parenthèses. Exemple : `maVariable=$(ls)` (plus lisible)

Récupération de la variable

- ❑ Utilisation avec le symbole \$ ou \${}
 - Exemple : `echo $maVariable`
 - Exemple : `echo ${maVariable}` (plus lisible)
- ❑ Affichage avec echo



« Les quotes »

- ❑ les apostrophes ou « simple quotes » : ''
- ❑ les guillemets ou « double quotes » : ""
- ❑ les accents graves « back quotes » : ``

Des comportements différents

- ❑ « simple quotes » -> la variable n'est pas affiché et le \$ est affiché tel quel
- ❑ « double quotes » -> demandent à bash d'analyser le contenu du message. S'il trouve des symboles spéciaux (comme des variables), il les interprète.
- ❑ « back quotes » -> toute la chaîne est interprétée.



commande read

Il est possible de demander à l'utilisateur de saisir du texte avec la commande read. Ce texte sera immédiatement stocké dans une variable.

Exemple : test.sh

```
read prenom  
echo "Votre prénom est : $prenom"
```

```
./test.sh
```

```
Filipe  
Votre prénom est : Filipe
```



commande read

Il est possible d'affecter simultanément une valeur à plusieurs variables avec read.

```
read nom prenom  
echo "Votre nom est $nom et votre prénom est $prenom"
```

read -p "Message"

```
read -p "Veuillez saisir votre nom et prénom :" nom prenom  
echo "Votre nom est $nom et votre prénom est $prenom"
```

D'autres options de read :

- n couper la chaîne de caractères Exemple : (5 caractères max)
- s masquer la chaîne saisie (mot de passe)
- t un temps de saisie

En bash, les variables sont toutes des chaînes de caractères. il n'est donc pas capable d'effectuer des opérations.

commande let

Les opérations +,-,*,/,**,%

```
let "a=5"
let "a+=3"
let "b=3"
let "c=a+b"
echo "a = $a b = $b c = $c"
```



`$()`

```
a=5
b=3
c=$((a+b))
echo "a = $a b = $b c = $c"
```

Pour les calculs sur des nombres décimaux, on utilisera la commande bc Exemple :

```
c=$(echo "scale=2;$a/$b" |bc)
```



Comme toutes les commandes, les scripts bash peuvent eux aussi accepter des paramètres. Ainsi, on pourrait appeler notre script comme ceci. Les différents paramètres sont séparés par des espaces.

```
./params.sh param1 param2 param3
```

Les valeurs de ces paramètres peuvent être récupérés dans le script en utilisant des variables prédefinies en bash

- \$# : Nombre de paramètres reçus
- \$0 : Nom du script utilisé
- \$* : Ensemble des paramètres reçus sous la forme d'une unique chaîne de caractères
- \$@ : Ensemble des paramètres reçus sous la forme d'un tableau
- \$1,\$2,\$3... : Désigne le i-ième paramètre reçu



Actuellement, les variables que vous créez dans vos scripts bash n'existent que dans ces scripts.

Les variables globales de env peuvent être utilisées dans un script.

Variables globales utiles (en majuscules)

- ❑ SHELL : indique quel type de shell est en cours d'utilisation
- ❑ PATH : liste des repertoires qui contiennent des exécutables que vous souhaitez pouvoir lancer sans indiquer leur répertoire.
- ❑ HOME : la position de votre dossier home.
- ❑ PWD : le dossier dans lequel vous vous trouvez.
- ❑ OLDPWD : le dossier dans lequel vous vous trouviez auparavant.

```
echo "Votre répertoire est :$HOME"
```