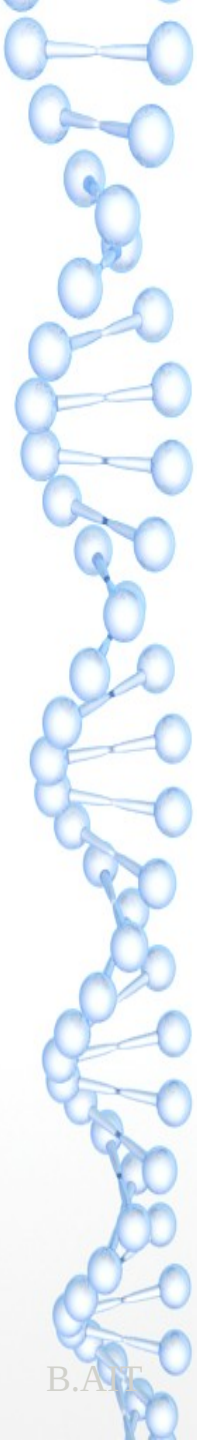

GNU/Linux

Introduction scripts Shell

Sommaire

Introduction scripts Shell

Le Shell

Définition :

Le shell est une interface texte qui permet à un utilisateur de communiquer avec le système d'exploitation.

Le shell sous GNU/Linux est à la fois un interpréteur de commandes et un langage de programmation.

L'intérêt du shell sous Gnu/Linux est qu'il permet de combiner les nombreuses commandes shell pour réaliser des actions complexes.

Les programmes écrits en langage shell s'appellent des **scripts shell** .

- Il existe différentes implémentions Shells, les plus courantes sont : sh (Bourne shell), bash(Bourne again shell), csh (C Shell), ksh (Korn shell), Zsh...

Le shell BASH, développé par GNU(sous licence GPL) est le Shell par défaut sous le système GNU/Linux.

Ce cours se basera sur le **Shell** BASH.

Introduction scripts Shell

Le Shell

Quelques caractéristiques du Shell

Type de commandes shell :

- **Les commandes internes** : Les commandes internes sont intégrées au programme Shell

Exemple :

La commande `cd` qui permet de se déplacer dans l'arborescence du système de fichier

n'a aucune existence sur le disque dur, elle est interne au Shell.

-**Les commandes externes :** Les commandes externes sont des fichiers localisés dans l'arborescence du système de fichier.

Exemple :

Lorsqu'on exécute la commande `ls`, le shell demande au noyau Linux de charger en mémoire RAM le fichier stocké dans `/bin/ls` .

Les commandes **internes** s'exécutent **plus rapidement** que les commandes **externes.**

Les commandes `which` et `type` permettent de vérifier si une commande est interne ou externe au Shell

Le Shell garde en mémoire les commandes exécutées
La commande `history` permet de lister l'historique des commandes exécutées.
(on remonte l'historique des commandes en appuyant sur la flèche vers le haut ou vers le bas du clavier).
Le Shell Auto-compléte une commande ou un nom de fichier lorsque vous appuyez sur la touche Tabulation.

Introduction scripts Shell

Créer un script SHELL

Pour créer un script shell, procédez comme suit ::

Exécutez l'éditeur de texte `nano` pour créer le fichier `script0.sh` qui contiendra liste des commande à exécuter.

Exemple :

```
$nano script0.sh
```

Placez la ligne suivante au début du fichier :

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

Le `#!` est appelé sha-bang.

cette ligne permet de s'assurer que le script est bien exécuté par le shell bash.

Les commentaires

- On peut ajouter des commentaires dans le script shell . Les commentaires sont des lignes qui ne seront pas exécutées mais qui permettent d'expliquer certaines commandes ou de les déactiver.

Tous les commentaires commencent par un `#`

Exemple :

```
#!/bin/bash

#Mon premier script bash script0.sh

# ce script affiche la liste des fichiers du répertoire courant

ls
```

Introduction scripts Shell

Créer un script SHELL

Exécuter un script Shell :

Pour pouvoir exécuter un script il doit avoir les droits d'exécution.

Exemple :

Pour exécuter `script0.sh` on procède comme suite:

```
$ chmod u+x script0.sh
```

Le script s'exécute maintenant comme n'importe quel programme, en tapant :

```
$./script0.sh
```

`"./"` devant le nom du script indique au shell que le fichier `script0.sh` est dans le répertoire courant .

Activer le débogage

pour résoudre certains problèmes liés à l'exécution des scripts, l'option `-x` de bash permet d'affiche le détail de son exécution comme ceci :

```
$ bash -x script0.sh
```

bash affiche alors le détail de l'exécution du script, ce qui peut aider à résoudre les erreurs :

Introduction scripts Shell

Les variables SHELL

Les variables Shell :

Une variable Shell sert à mémoriser une information temporairement pendant l'exécution d'un script ou d'une commande.

Une variable Shell est définie comme suite :

Syntaxe :

```
NOMVAR=valeur
```

Pour traiter le contenu d'une variable on precede celle-ci par le symbole $

Par exemple pour afficher le contenu d'une variable on exécute la commande :

```
 echo $NOMVAR
```

On peut supprimer une variable avec la commande :

```
unset NOMVAR
```

On distingue de types de variables :

- variable locale qui n'est disponible que pour le Shell dans lequel elle a été créée.

- **variable d'environnement** qui est disponible pour le Shell dans lequel elle a été créée et aussi dans les autres Shell .

Les commandes `set` et `env` affichent la liste des variables.

Introduction scripts Shell

Les variables SHELL

Exemples de variables d'environnement shell

Les variables positionnelles

lorsqu'une commande est exécutée par le Shell, celle-ci est stockée dans une variable nommée 0 et les arguments qui suivent la commande sont stockés dans les variables 1, 2, 3, etc.

La variable `PATH`

définie le chemin de recherche des commandes exécutées par le Shell.

La commande `echo $PATH` permet d'afficher sont contenu.

La variable code retour d'une commande «?»

Pour la mise au point d'un script, la variable «**?**» stock le code retour de la dernière commande exécutée.

`$?=0` si la commande s'est exécuté avec succès (vraie).

`$?` est supérieur à `0` si la commande échoue (fausse).

Les structure de contrôle

Il existe une parenté entre le langage Shell et les principaux langages algorithmiques comme le langage C.

Le Shell est capable d'exprimer des expressions conditionnelles complexes.

La structure `if then else fi`

La syntaxe:

```
if condition
then commandes si condition vraie
else commandes si condition fausse
fi
```

- La structure de contrôle `if` permet de réaliser des tests.
- La `condition` située à droite du `if` est exécutée.
- Si le code retour de la `condition ($?)` est égal à 0 (vrai), les commandes situées dans le bloc `then` sont exécutées.
- Si le code de retour est supérieur à 0 (faux), ce sont les commandes situées dans le bloc `else` qui sont exécutées.

Les structure de contrôle

La structure `case in esac`

- Quand il y a un nombre important de choix, la structure `case` est plus appropriée que la structure `if`.

Syntaxe :

```
case $var in

a)traitement si $var contient "a" ;;

b|c)traitement si $var contient "b" ou "c";;

d*)traitement si $var commence par "d";;

*)traitement par défaut;;

esac
```

La structure de contrôle `case` permet d'orienter la suite du programme en fonction d'un choix de différentes valeurs.

Les structure de contrôle

La structure `for in do done`

- La boucle for permet de traiter une liste de valeurs indiquée à droite du mot clé in.
- A chaque tour de boucle, la variable var est initialisée avec une des valeurs de la liste.
- Elles sont traitées entre le **do** et le **done** par la `suite d'instructions` dans l'ordre de leur énumération.

Syntaxe :

```
for var in liste de valeurs

do

suite d'instructions

Done
```

var prend successivement pour valeur chaque argument de la `liste de valeurs`.

Les structure de contrôle

La structure `while do done`

La boucle `while` permet d'exécuter les commandes présentes entre le **`do`** et le **`done`** tant que la condition placée à droite du **`while`** retourne un code retour vrai.

La syntaxe :

```
while condition
do
commandes tant que la condition est vraie
done
```

Les structure de contrôle

Les opérateurs logiques `&&` et `||`

`&&` (le et logique) et `||` (le ou logique) sont des séparateurs de commandes conditionnels.

Syntaxe :

```
commande1 && commande2
```

Le shell exécute `commande2` si `commande1` est exécutée sans erreur.

```
commande1 || commande2
```

Le shell exécute `commande2` si `commande1` est exécutée avec erreur.

Les structure de contrôle

La commande `break`

La commande `break` permet de sortir d'une boucle d'une manière inconditionnelle

La commande `read`

La commande `read` lit son entrée standard et affecte les valeurs saisies dans la ou les variables passées en argument.

Introduction scripts Shell

Les Fonctions et les alias SHELL

Fonction :

Une fonction shell est un ensemble de commandes identifiées par un nom de fonction et résidant en mémoire principale(RAM). Un script peut être composé de plusieurs fonctions chacune réalisant une tâche particulière. Cette modularité permet d'accélérer l'exécution du script et permet aussi de simplifier les tâches de mises au point et mises à jour des scripts shell.

Syntaxe :

```
nom_foction (){ liste de commandes ;}
```

Exemple :

```
recherche (){ find $HOME –name s1 ;}
```

Alias :

L'alias est le mécanisme qui permet de designer une commande ou une suite de commandes par un nouveau nom.

Le alias sont souvent utilisées pour alléger la syntaxe d'une commande

Syntaxe :

```
alias nouvelle_commande='chaine'
```

Exemple :

```
alias list= 'ls   –ail'
```

Introduction scripts Shell

Un script Shell bien écrit vaut mieux qu'un millier de commandes tapées à la main.

Un bon script est comme un assistant silencieux qui travaille pour vous.