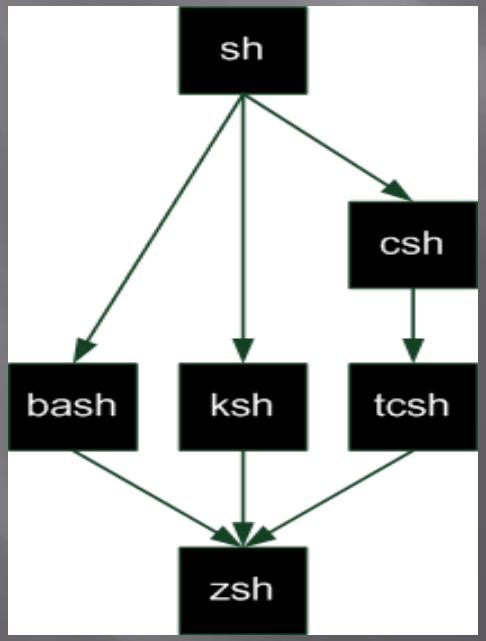
LES SCRIPTS SHELL

BENNIS Naim
BRUEL Jean-François
LY Boubacar

Plan

1. Les Shells
1.1. Qu'est ce qu'un shell ?

- Un interpréteur de commande
- Installé sur tous les OS basé sur UNIX
- Sous Windows le programme analogue est Commande.Com
- Les différents shells :



1. Les Shells
1.2. A quoi servent-ils ?

- Les shells permettent par exemple de:
  - gérer l'invite de commande
  - gérer l'auto-complétion
  - gérer les dernières commandes utilisées
  - gérer les processus
  - rediriger et chaîner les commandes (symboles >, <, |, ...))

1. Les Shells
1.3. Comment les installés ?

- Sh est toujours installé sur les OS basé sur UNIX

- Pour les autres on utilise la méthode comme pour d'autres paquets :
  apt-get install nom_du_shell

  - Exemple : apt-get install ksh

1. Les shells
1.4. Relations avec les scripts shell

- Les commandes du scripts sont interprétées par le shell
- Selon le shell installé les commandes utilisées peuvent varier
- Nous allons donc vous présenter Bash
  - On le trouve par défaut sous Linux et Mac OS X
  - Il rend l'écriture de scripts plus simple que sh.
  - Plus répandu que ksh, zsh

2. Les Scipts Shells
2.1 Qu'est ce qu'un script shell?

- Le scripting shell est un mini-langage de programmation
  - Il permet d'automatiser des tâches répétitives
  - On peut ainsi créer ses propres tâches
- Il se présente sous la forme d'un fichier
  - Contient une ou plusieurs commandes exécutées séquentiellement
- Langage interprété

2. Les Scripts Shells
2.2. Créer et exécuter un script

- 2 méthode pour créer un script
  - Taper dans un shell toutes les commandes
  - Rassembler toutes les instructions dans un fichier
    - Le script débute par: #!/bin/bash
    - Il se termine de préférence par: exit 0;

- 2 méthode pour exécuter un script
  - bash nom_du_script
  - chmod +x nom_du_script puis ./nom_du_script

2. Les Scripts Shells
2.3. Les commandes

- Les principales commandes:
  - echo : permet d'afficher ses paramètres
  - ls : permet de lister le contenu du répertoire courant
  - grep : permet d'extraire certaine ligne de fichiers
  - read : permet de remplir des variables
  - exec :
    - exec *commande*: remplace le script shell par la *commande*
    - exec *redirection*: applique *redirection* indiqué au shell courant
  - cp ,date, pwd, man ,....

2. Les Scripts Shells
2.4. Les variables et les fonctions

- Les variables
  - Créer ses variables
    - Déclaration : MSG=salut
    - Utilisation : echo $MSG
  - Variables des paramètres
    - $* contient tous les arguments passés à la fonction
    - $# contient le nombre d'argument
    - $? contient le code de retour de la dernière opération
    - $0 contient le nom du script
    - $n contient l'argument n, n étant un nombre

2. Les Scripts Shells
2.4. Les variables et les fonctions

- Les variables
  - Les variables d'environnements
    - **SHELL** : indique quel type de shell est en cours d'utilisation (sh, bash, ksh...)
    - **PATH** : une liste des répertoires qui contiennent des
    - **EDITOR** : l'éditeur de texte par défaut qui s'ouvre lorsque cela est nécessaire.
    - **HOME** : la position de votre dossier home.
    - **PWD** : le dossier dans lequel vous vous trouvez.
    - **OLDPWD** : le dossier dans lequel vous vous trouviez auparavant

2. Les Scripts Shells
2.5. Les variables et les fonctions

- Créer des fonctions pour des actions précises :
  - Création : nom_de_la_fonction() { liste d'instruction }
  - Appel : nom_de_la_fonction $1 $2 $3
  - Exemple:

```
#!/bin/sh
#Definition d'une fonction
mafonction(){
    echo 'La liste des fichiers de ce répertoire‘
     ls -l
}
echo 'Vous allez voir la liste des fichiers de ce répertoire:'
#appel de ma fonction
mafonction
exit 0
```

2. Les Scripts Shells
2.5. Les structures de contrôles

- Il existe 3 types de tests différents en bash :
  - Tests sur des chaînes de caractères
  - Tests sur des nombres
  - Tests sur des fichiers

- **Tests sur des chaînes de caractères**
  - $chaine1 = $chaine2
  - $chaine1 != $chaine2
  - -z $chaine
  - -n $chaine

2. Les Scripts Shells
2.5. Les structures de contrôles

- Exemple

```
if [ -z $1 ]
then
   echo "Pas de paramètre"
else
   echo "Paramètre présent«
fi
```

2. Les Scripts Shells
2.5. Les structures de contrôles

- **Tests sur des nombres:**
  - $num1 -eq $num2 → =
  - $num1 -ne $num2 → !=
  - $num1 -lt $num2 → <
  - $num1 -le $num2 → <=
  - $num1 -gt $num2 → >
  - $num1 -ge $num2 → >=

2. Les Scripts Shells
2.5. Les structures de contrôles

- **Tests sur des fichiers**
  - -e $nomfichier → Teste si le fichier existe
  - -L $nomfichier →Teste si le fichier est un lien symbolique
  - -d $nomfichier →Teste si le fichier est un répertoire
  - -f $nomfichier →Teste si le fichier est un fichier
  - -w $nomfichier →Teste si le fichier est modifiable
  - -x $nomfichier →Teste si le fichier est exécutable
  - -r $nomfichier →Teste si le fichier est lisible
  - $fichier1 -nt $fichier2 → **n**ewer **t**han
  - $fichier1 -ot $fichier2 → **o**lder **t**han

2. Les Scripts Shells
2.5. Les structures de contrôles

- If
  - Le type de condition le plus courant est le **if**, qui signifie "si".
  - structure

```
if [ test ]; then
effectuer_une_action1
elif [ encore_autre_test ]
then
 effectuer_une_action2
Else
effectuer_une_action1
fi
```

  - Exemple:

```
if [ $1 = « gestion_projet" ]
 then
            echo  "Exposé matin !"
elif  [ $1 = « java" ];
then
            echo " Pas exposé"
else
            echo « rien »
fi
```

2. Les Scripts Shells
2.5. Les structures de contrôles

- While
  - Permet de faire prendre un ensemble de valeurs à une ou plusieurs variables
  - Pour chaque valeurs les commandes entre do et done sont exécutées
  - Exemple:

```
while [ "$mdp" != "ubuntu" ] && [ "$cmpt" != 4 ]
do
  echo -n «Mauvais mot de passe, plus que"$cm" chance(s) »
  read mdp
  cmpt=$(($cmpt+1))
  cm=$(($cm-1))
done
```

2. Les Scripts Shells
2.5. Les structures de contrôles

- Case
  - Permet d'exécuter une suite différente de commandes en fonction du paramètre
  - Exemple :

```
read on
case "$on" in
   oui | o | O | Oui | OUI ) echo « oui!";;
   non | n | N | Non | NON ) echo « non !";;
   * ) echo « ni oui, ni non »;;
esac
```

2. Les Scripts Shells
2.5. Les structures de contrôles

- For
  - Permet de faire prendre un ensemble de valeurs à une variable
  - Pour chaque valeurs les commandes entre do et done sont exécutées.
  - Exemple:

```
for VARIABLE in toto tutu « ti ti»;
do
  echo $ VARIABLE;
done
```

Conclusion

- Difficile de présenter les Scripts en 15 min
- Bibliographie
  - http://doc.ubuntu-fr.org/tutoriel/script_shell#la_commande_test
  - http://www.siteduzero.com/tutoriel-3-123626-afficher-et-manipuler-des-variables.html

Avez-vous des questions ?