Introduction à la ligne de commande 3
« Automatisation ou comment éviter de faire deux fois la même chose »

Louvilug

28/10/2010

Syntaxe de **find**

Synopsis

```
find [répertoire] [critères] [action]
```

Rappel sur les critères

- **-name** [motif] : cherche les fichiers dont le nom contient [motif] .
- **-iname** [motif] : même chose, mais sans tenir compte des majuscules/minuscules.
- **-type** [caractère] Restreint la recherche à un type de fichiers en fonction de [caractère] :
  - **f** un fichier régulier.
  - **d** un répertoire (directory).
  - **l** un lien.
- **-user** [motif] : cherche les fichiers dont le nom du propriétaire contient [motif] .
- `man find` pour une liste exhaustive . . .

Actions prédéfinies

- **-print** : l'action par défaut : affiche le nom du fichier.
- **-ls** : affichage similaire à `ls -l` .
- **-delete** : détruit le fichier après l'avoir trouvé.
- À nouveau, `man find` pour une liste exhaustive . . .

Actions personnalisées

Structure

```
-execdir [commande]   [arguments]   [ '{}' ]   ';'
```

Actions personnalisées

Structure

```
-execdir [commande]   [arguments]   [ '{}' ]   ';'
```

Exemple

- `find -type d -execdir chmod 755 '{}' ';'`

Actions personnalisées

Structure

```
-execdir [commande]   [arguments]   [ '{}' ]   ';'
```

Exemples

- `find -type d -execdir chmod 755 '{}' ';'`
- `find -iname '*.jpg' -execdir mogrify '{}' -resize '64x64>' ';'`

Actions personnalisées

Structure

```
-execdir [commande]   [arguments]   [ '{}' ]   ';'
```

Exemples

- `find -type d -execdir chmod 755 '{}' ';'`
- `find -iname '*.jpg' -execdir mogrify '{}' -resize '64x64>' ';'`
- `find -name '*.mp3' -execdir sox '{}' "$(echo {}|sed 's/mp3$/ogg/')" ';'`

À quoi sert un script ?

On voudrait **sauver** une série de commandes, et les **éxécuter** sans les retaper une à une dans le shell.

syntaxe d'un script bash

```
# Les commentaires sont sur les lignes commencant
# par un diese ('#')
```

syntaxe d'un script bash

```
#!/bin/bash

# La ligne ci-dessus DOIT etre la premiere ligne
# fichier script
```

syntaxe d'un script bash

```
#!/bin/bash

# Par exemple
apt-get -y update
apt-get -y install ssh vim nethack
halt
#script a executer en tant que root
```

Éxecuter le script

- S'assurer que le droit d'éxécution est mis :

```
chmod a+x monscript.sh
```

Éxecuter le script

- S'assurer que le droit d'éxécution est mis :
  `chmod a+x monscript.sh`
- En tant que processus indépendant :
  `./monscript.sh` dans le répertoire où se situe le script

Éxecuter le script

- S'assurer que le droit d'éxécution est mis :
  `chmod a+x monscript.sh`
- En tant que processus indépendant :
  `./monscript.sh` dans le répertoire où se situe le script
- Dans le shell courant :
  `source monscript.sh` toujours dans le répertoire courant

Variables

Toute les variables sont des chaines de caractères !

- Déclaration :
  `[nom]  =  [valeur]`
- Substitution :
  `$[nom]`

Arguments d'un script

- `$1` : Le premier argument.
- `$2` : Le second argument.
- ...

Arguments d'un script

- `$0` : Le chemin du fichier du script, par exemple `/home/moi/monscript.sh`
- `$1` : Le premier argument.
- `$2` : Le second argument.
- ...

Arguments d'un script

- `$0` : Le chemin du fichier du script, par exemple `/home/moi/monscript.sh`
- `$1` : Le premier argument.
- `$2` : Le second argument.
- ...
- `$#` : le nombre d'arguments.

Arguments d'un script

- `$0` : Le chemin du fichier du script, par exemple `/home/moi/monscript.sh`
- `$1` : Le premier argument.
- `$2` : Le second argument.
- ...
- `$#` : le nombre d'arguments.
- `$@` : la liste de tous les arguments.

Conditions

```
 if [ [test]  ]
then
    [instruction(s)]
elif [ [test2]  ]
    [instruction(s)]
else
    [instruction(s)]
end
```

Conditions

```
 if [ [test]  ]
then
    [instruction(s)]
elif [ [test2]  ]
    [instruction(s)]
else
    [instruction(s)]
end
```

Conditions

```
 if [ [test1]  ]
then
    [instruction(s)]
elif [ [test2]  ]
    [instruction(s)]
else
    [instruction(s)]
end
```

Conditions

- `-n [chaine]`
  Vrai si `[chaine]` est non-vide.
- `-z [chaine]`
  Vrai si `[chaine]` est vide.
- `[ch1] == [ch2]`
  Vrai si `[ch1]` et `[ch2]` sont les mêmes chaines.
- `-e [chaine]`
  Vrai si le fichier existe.
- `-x[chaine]`
  Vrai si le fichier existe et est exécutable.
- `man bash` à la section « Conditional Expressions » pour tous les tests . . .

Exemple

```
# !/bin/bash

#set message
if [ -z "$1" ]
then
    msg='Hello, world !'
else
    msg="$1"
fi

#set output file
if [ -z "$2" ]
then
    file=output_file
else
    file="$2"
fi

#output message
current_date=$(date '+%d:%m:%Y,%M:%H:%S')
echo "$current_date $msg" >> $file
```

Qu'est-ce que **cron** ?

Cron est un daemon qui exécute régulièrement des tâches spécifiées par un utilisateur.

Lancer cron

Cron devrait se lancer au démarrage de l'ordinateur.
Sinon, pour lancer cron, tapez en tant que root :

```
/etc/rc.d/crond start
```

Comment éditer le crontab ?

Éditer le crontab de l'utilisateur `crontab -e`

syntaxe du crontab utilisateur

```
#commentaire identique au script bash
#Les variables d'environnement sont declarees ainsi :
variable=valeur

#minute heure jour_mois mois jour_semaine commande
0,30    12    *          *    1-5          monscript.sh $var
```