Apprendre à rédiger des scripts sous bash

- Objet : Apprendre à rédiger des scripts sous bash
- Niveau requis :
  débutant, avisé
- Commentaires : Fix Me!
- Débutant, à savoir : Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Suivi :
  - Création par Hypathie le 18/03/2014
  - Testé par Hypathie le Juin 2014
- Commentaires sur le forum : Lien vers le forum concernant ce tuto [1)]

Contributeurs, les Fix Me! sont là pour vous aider, supprimez-les une fois le problème corrigé ou le champ rempli !

Nota : Les autres wiki :

-
- script-bash-variables-arguments-parametres
- script bash : modification de variable et de paramètre
- script-bash-enchainement-de-commandes-et-etat-de-sortie
- script-bash-etat-de-sortie-et-les-tests
- script-bash-les-tableaux
- script-bash-les-fonctions

Introduction : éviter les bashismes

Shell, Path, Bash, commande : quelques rappels !

Ré-requis indispensables :

- Utiliser GNU/Linux en ligne de commande, tout commence là !.
- Le shell pour tous
- Bash : Introduction

- pour s'exercer sur la question de chemin relatif et absolu : illustration-navigation-shell
- un résumé : caractères symboliques

La norme POSIX et l'étude des scripts Bash

Vous savez donc ce qu'est le shell, un alias et un script.
Mais quel rapport entre la diversité des shell (ou interpréteur de commandes) qui existent (sh ; bsh ; bash ; ksh, etc.) et les scripts ?

- C'est qu'avec l'en-tête du script, chacun des **sha-bang** ci-dessous appelle un interpréteur de

commandes différent:

```
#!/bin/sh
#!/bin/bash
#!/bin/perl
#!/bin/tcl
```

- POSIX[2] est un standard :

Utiliser `#!/bin/sh` permet de tenir compte du standard sh de POSIX.

Voici un PDF assez complet pour apprendre à utiliser le shell sh : http://igm.univ-mlv.fr/~masson/Teaching/PIM-INF3/shell.pdf

- Appeler bash avec l'option `–posix` ou insérer `set -o posix` au début du script fait que bash se conforme au standard posix.

À savoir :

- Shell compatibles avec sh : bash, ksh
- Shell incompatibles avec sh : csh, tcsh

Mieux vaut apprendre à écrire des scripts Bash en connaissance de cause en ce qui concerne la norme POSIX !
Et cela même si le Shell par défaut est le Bash, sur la plupart des distributions Linux.

Pour utiliser tcsh, ksh, ash, sh, csh, *etc* :
http://formation-debian.viarezo.fr/shell.html
méthode d'installation de ksh

Apprendre le Bash sans devenir un “ultra-bashiste” : si si c'est possible ! 😎

Écrire des scripts Bash POSIX

En général, tous les shell acceptent la même syntaxe de base telle que définie par POSIX, mais chacun accepte une syntaxe étendue qui lui est propre (et donc incompatible avec les autres shells).

Voici quelques aspects auxquels se référer à chaque fois que vous apprendrez une nouvelle notion relative au shell Bash.

- D'abord, un tableau récapitulatif qui met en avant la question de la syntaxe POSIX :

POSIX	**À éviter : bashisme**
if [“$toto” = “$titi”] ; then …	if [“$toto” == “$titi”] ; then …

POSIX	**À éviter : bashisme**
diff -u `fichier.orig fichier.c`	diff -u `fichier.{orig,c}`
mkdir `/tototiti /tototutu`	mkdir `/toto{titi,tutu}`
`funcname() { ... }`	`function funcname() { ... }`
format octal : « \377 »	format hexadécimal : « \xff »

- Attention à la commande `echo`, ses options ne sont pas prises en compte de la même manière selon les shell :
  1. Éviter l'utilisation des options de commande `-e` et `-E`.
  2. Éviter d'utiliser toutes les options de commandes sauf `-n`.
  3. Éviter d'utiliser les séquences d'échappement dans les chaînes de caractères car leur prise en compte varie.
  4. Utilisez la commande `printf` plutôt que la commande `echo` si vous avez besoin d'intégrer des séquences d'échappement dans la chaîne de sortie.

Yep ! Utiliser printf n'est pas difficile !

- Syntaxe:

```
printf format [argument]....
```

- Options basiques :
  - `\b` : Espace arrière
  - `\n` : Nouvelle ligne
  - `\t` : Tabulation horizontale
  - `\v` : Tabulation verticale.
- Exemple :

```
printf "Coucou\n"
```

retour de la commande

```
Coucou
```

- Pour aller plus loin :
  - `man printf`
  - Explications illustrées d'exemples : The printf command

- Enfin, même s'il n'est pas question pour un débutant d'intégrer tout ce qui suit, voici les avantages et les particularités propres au shell Bash :
  1. Certaines options étendues d'appel
  2. La substitution de commandes utilisant la notation `$( )`
  3. Certaines opérations de manipulations de chaînes
  4. La substitution de processus
  5. Les commandes intégrées de Bash

Pour plus de détails sur chacun de ces points voir : Guide avancé d'écriture des scripts Bash :36.9. Problèmes de portabilité

Il s'agit là plutôt d'un aboutissement, essayons d'acquérir par des exemples très simples, les connaissances de bases qui permettront de comprendre chacun de ces points, ainsi que ce que l'on trouve ici : scripts

Comment créer et exécuter un script ?

Il y a différentes méthodes pour lancer ses scripts, cela dépend, vous l'aurez compris, du répertoire dans lequel sont placés ses scripts.

- Commençons par créer un script nommé mon-script :

```
touch mon-script
```

Si l'on a exécuté cette commande à l'ouverture de son terminal, le fichier “mon-script” est alors placé dans son répertoire courant.
Et oui un script est un simple fichier texte dont le contenu (une suite de commandes et d'instruction) est exécutable.

- Puis donnons à ce fichier les droits d'exécution :

À savoir :

- droits-unix
- droits-unix-bis

```
chmod u+x mon-script
```

Voyons maintenant trois méthodes pour exécuter un script

"bash nom-script"

- Éditons le fichier “mon-script” par exemple nano

```
nano mon-script
```

dans lequel on inscrit :

mon-script

```
echo -n "Bonjour les copains"
```

- Pour exécuter ce script il suffit d'inscrire dans un terminal.

```
bash nom-du-script
```

> Comme on le fait pour une commande.

Attention, il faut penser à se déplacer dans le répertoire parent de ce script avant de lancer l'exécution.

```
bash mon-script
```

retour de la commande

```
Bonjour les copainsutiliateur@debian:~$
```

Pour avoir un comportement standard avec la commande echo, on peut aussi écrire ainsi :

```
/bin/echo -n "Bonjour les copains"
```

> La commande echo est une commande interne du shell ; la commande /bin/echo est une commande à part.

merci captnfab

Essayez maintenant :

```
bash --posix mon-script
```

retour de la commande

```
Bonjour les copainsutilisateur@debian:~$
```

Le sha-bang et ./mon-script

- Reprenons notre fichier “mon-script” (avec cette fois un autre programme, celui de la commande ls par exemple).

mon-script

```
#!/bin/bash
ls /home/utilisateur
```

- Vous pouvez maintenant exécuter le fichier exécutable “mon-script” en faisant :

```
./mon-script
```

Il faut là aussi se trouver dans le répertoire parent du script pour l'exécuter de cette façon.
Mais il est à noter que l'écriture `./fichier` désigne un fichier exécutable, donc des scripts.
Dans les wiki de façon générale, lorsqu'il s'agit d'un script on utilise `./fichier` afin de savoir de quoi on parle, sans avoir à se soucier de la configuration du fichier `~/.bashrc` des lecteurs.

Exécuter son script depuis n'importe où !

À voir : modifier-durablement-la-valeur-de-la-variable-d-environnement-path

- Il faut pour cela placer le chemin absolu de son script dans le Path,
  C'est-à-dire dans l'un des répertoires `/bin`, `/usr/bin` ou `/usr/local/bin`
- mais à notre niveau, les scripts que l'on crée sont ceux de l'utilisateur.
  On peut donc simplement ajouter le chemin du répertoire dans lequel on range ses scripts :

1. en éditant le fichier `~/.bashrc` qui est un fichier caché du répertoire courant de l'utilisateur (son répertoire personnel);
2. et en y ajoutant, par exemple en dernière ligne, `PATH=$PATH“:$HOME/MesScripts`.

Application

- Créons un fichier de type répertoire nommé par exemple “MesScripts” :

```
mkdir MesScripts
```

- Puis éditons `~/.bashrc`

```
nano ~/.bashrc
```

- Pour y ajouter, en dernière ligne : `PATH=$PATH”:$HOME/MesScripts“` .

Sans éditer, le fichier `~/.bashrc`, on peut y ajouter cette ligne en faisant :

```
echo 'PATH=$PATH":$HOME/MesScripts"' >> ~/.bashrc
```

Merci à phlinux pour cette remarque

- Créons le fichier “mon-script” avec le sha-bang :

```
nano mon-script
```

- Contenant les lignes :

mon-script

```
#!/bin/bash
printf "yep coucou!\n"
```

- Donnons à “mon-script” les droits d'exécution :

```
chmod u+x ~/mon-script
```

- et plaçons “mon-script” dans le répertoire MesScripts (dont le chemin est ajouté au PATH) :

```
mv ~/mon-script ~/MesScripts/
```

- Il ne reste plus qu'à réinitialiser le shell Bash, en fermant puis en ré-ouvrant son terminal, ou en exécutant l'une de ces deux commandes équivalentes :

```
source ~/.bashrc
```

```
. ~/.bashrc
```

Ou encore

```
exec $SHELL
```

> Et voilà pour exécuter son script, il suffit maintenant de tapez dans le terminal le nom de son script !

```
mon-script
```

doc:programmation:shells:la-page-man-bash-les-caracteres-speciaux yep coucou! </file>

- Vous pouvez voir maintenant votre répertoire dans la liste des répertoires du Path.

Fermez et ré-ouvrez le terminal; puis tapez :

```
echo $PATH
```

doc:programmation:shells:la-page-man-bash-les-caracteres-speciaux /usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/utilisateur/MesScripts </file>

- Concernant la commande `source ~/.bashrc`\\(ou son équivalent `. ~/.bashrc`):
  notons que cela ne fonctionne que pour faire prendre en compte un ajout dans le fichier `~/.bashrc`.
  Si au contraire, on dé-commente une ligne de ce fichier, il faudra fermer puis ré-ouvrir le terminal pour que le changement soit effectif.

Pas si difficile que ça

Un petit script pour lancer un script depuis n'importe où !

Pas d'inquiétude si vous ne comprenez pas tout ; vous en en serez capable après avoir suivi les wiki sur les scripts, et consulté leurs liens. Il faut :

1. avoir créé un fichier de type répertoire (ex: MesScripts) ;
2. avoir modifié le fichier `~/.bashrc` pour ajouter au Path le chemin de son fichier de type répertoire (MesScripts) (comme ci-dessus);
3. avoir fermé le terminal et l'avoir ré-ouvert ;
4. avoir créé un fichier (ex: nommé ici scriptx) ;
5. avoir donné à l'utilisateur les droit d'exécution sur “scriptx” ;
6. avoir placé “scriptx dans le répertoire MesScripts” ;
7. ouvrez ce fichier “scriptx” et collez-y le code ci-dessous ;
8. enregistrez et lancez-le depuis un terminal.

scriptx

```
#!/bin/bash
set -o posix
printf "Un nouveau script $USER ? Son nom : "
{ read nom ; echo "#!/bin/bash" >> $nom ; chmod u+x $nom ; mv ~/$nom
~/MesScripts ; /usr/bin/gedit ~/MesScripts/$nom ;}
```

À savoir : une suite de commandes s'écrit de façon équivalente pour le shell ainsi :

script

```
#!/bin/bash
set -o posix
printf "Un nouveau script $USER ? Son nom : "
{ read nom ;\
 echo "#!/bin/bash" >> $nom ;\
 chmod u+x $nom ;\
 mv ~/$nom ~/MesScripts ;\
 /usr/bin/gedit ~/MesScripts/$nom ;}
```

> Espace avant le point virgule, puis antislash \ accolé au point virgule ; puis ↵ Entrée (espace ou non avant la nouvelle commande).

Merci à captnfab et LeDub pour cette information !

- Pour le lancer :

```
scriptx
```

retour de la commande

```
Un nouveau script toto ? Son nom :
```

> Lors de l'exécution de ce script, la chaîne que vous entrerez pour répondre à la question, sera le nom du nouveau script que vous voulez créer.

Astuces

Imaginons que nous avons beaucoup de scripts et que vous souhaiter les ranger dans différents dossiers pour qu'ils soit classés.
Pour lancer un script (d'essai par exemple sans se soucier s'il est rangé dans l'un de vos différents répertoires indiqués dans le fichier `~/bashrc`.
On peut éditer ce dernier, et y ajouter

- soit cette ligne : `PATH=$PATH":."`
- soit cette ligne : `PATH=$PATH":$HOME/."`

> Dans ces deux cas, quelque soit l'endroit où l'on se trouve dans l'arborescence, on peut lancer son script en l'appelant avec son nom !

la suite c'est ici

script-bash-variables-arguments-parametres

1)
N'hésitez pas à y faire part de vos remarques, succès, améliorations ou échecs !
2)
Portable Operating System Interface.
Voir :
http://fr.wikipedia.org/wiki/POSIX
http://polytechnice.free.fr/Archives/SI/SI3/Systeme/Cours/posix.pdf
Un lien indispensable en anglais : http://hyperpolyglot.org/shell
Les spécifications posix sont disponibles sur le site :
http://pubs.opengroup.org/onlinepubs/007904975/toc.htm
Pour décortiquer les bashismes : http://rgeissert.blogspot.fr/search/label/bashisms **Merci** **captnfab** 😉

From:
http://debian-facile.org/ - **Documentation - Wiki**

Permanent link:
http://debian-facile.org/doc:programmation:shells:debuter-avec-les-scripts-shell-bash

Last update: **20/08/2022 08:35**