

0.1 Agrémenter son shell

Le Shell est un interpréteur de commandes. Lorsque vous tapez une commande, il effectue différentes opérations et transfère la commande interprétée au noyau du système d'exploitation, pour exécution. Dans le cas où la commande est simple, il suffit de la taper directement après le prompt. Mais parfois une commande nécessite plusieurs arguments, des tests et des redirections, auquel cas il est préférable d'écrire un Programme¹. Si un interprète de langage de commandes bien précis s'exécute au moment de la connexion de l'utilisateur au système, il n'y est pas forcément lié. L'utilisateur peut à tout moment décider de changer d'interprète shell que ce soit d'une manière temporaire ou définitive. En général, plusieurs interprètes shell sont disponibles sur le système. Rappelons que les interprètes de commandes ne sont rien d'autre que des utilitaires comme les autres contenus sur disque dans des fichiers binaires réguliers. Le plus connu sous linux est bash mais il y'a aussi zsh, ksh, csh ... Je me baserai ici sur bash qui est très performant et qui de plus, est de loin le plus courant sous linux, évidemment beaucoup de ces principes sont transposables à d'autres shells moyennant quelques ajustements.

0.1.1 1. Enrichir le PATH de tous les utilisateurs :

Le Path est une variable d'environnement, c'est en fait la liste des répertoires contenant des exécutables. Pour exécuter un programme vous devez taper le chemin exact vers l'exécutable. Par exemple, pour exécuter mozilla vous devrez probablement taper /usr/bin/mozilla. Cela peut devenir très fastidieux surtout qu'il est difficile de se souvenir de l'emplacement de tous les exécutables (/sbin, /usr/sbin, /bin ...). La solution la plus élégante est de mettre ces répertoires récurrents dans votre path. Rassurez-vous cependant, votre distribution en s'installant définit un path minimum, pour tous les utilisateurs, c'est ce qui vous permet de taper ls au lieu de /usr/bin/ls, dans un terminal. Vous pouvez voir le contenu de votre path par : **echo \$PATH**, chez moi j'obtiens : /usr/local/bin : /bin : /usr/bin : /usr/local/bin : /usr/bin : /usr/gnat/bin : . : /opt/www/htdig/bin : /usr/lib/j2re1.4.0_01/bin : /opt/kde/bin : /usr/lib/qt-3.0.4/bin : /sbin Notez les " : " qui sont des séparateurs de répertoires, notez aussi le "." , c'est le répertoire courant, cela permet d'exécuter un programme dans le répertoire courant par le_prog au lieu de ./le_prog

0.1.2 1.1 Modification globale du path :

Il est possible de modifier globalement le path, c'est à dire pour tous les utilisateurs en éditant le fichier /etc/profile. Pour rajouter par exemple le répertoire /sbin au path global, rajoutez les lignes suivantes en fin de fichier :

```
PATH=$PATH :/sbin
export PATH
```

Pour que les modifications soient prises en compte pour vous à la volée, tapez :

```
source /etc/profile
```

Sinon au prochain démarrage, se sera le cas.

¹ <http://www.trustonme.net/didactels/148.html>

0.1.3 1.2 Modification locale du path :

Le path de chaque utilisateur est composé de l'intégralité du path global, auquel vous pouvez rajouter quelques répertoires personnels. Pour se faire vous devez éditer le fichier de configuration local. Généralement c'est le fichier `~/bashrc`, pour certaines distributions comme Mandriva ou Slackware, le plus propre serait d'utiliser `~/bash_profile`. Les modifications à apporter sont rigoureusement les mêmes, il s'agit de rajouter la ligne suivante à la fin du dit fichier (ou de la modifier si elle existait déjà) :

```
PATH=$PATH :/sbin :/usr/jeux/hl
```

Ceci supposant que vous souhaitiez rajouter `/usr/jeux/hl` à votre path. Pour que les modifications soient prises en compte, fermez le terminal et ouvrez en un autre.

0.1.4 2. Les Aliases :

Les aliases permettent de définir des raccourcis vers d'autres commandes. Ils peuvent affecter tous les utilisateurs si le root les définit dans `/etc/profile` ou un seul utilisateur si ce dernier les définit dans `~/bashrc` (ou `~/bash_profile`). La syntaxe est la suivante : `alias la_commande_qui_sera_tapée='la_commande_qui_sera_exécutée'`. Quelques alias courants :

– `cd..`

```
alias cd..='cd ..'
```

quand vous taperez `cd..` se sera `cd ..` qui s'exécutera

– `ls` en couleurs + quelques gadgets :

```
export LS_OPTIONS='--color=auto'
```

```
eval 'dircolors'
```

```
alias ls='ls $LS_OPTIONS'
```

```
alias ll='ls $LS_OPTIONS -lh'
```

```
alias l='ls $LS_OPTIONS'
```

– `su` avec `xhost+` :

```
alias su='xhost + ; su && xhost -'
```

Sur certaines distributions comme Slackware ou Debian, lorsque vous vous êtes logué en root par `su`, vous n'avez pas le droit d'exécuter des applications graphiques, sauf bien-sûr en tapant `xhost+` (avant `su`). Une façon efficace d'automatiser les choses est de rajouter la ligne ci-dessus.

0.1.5 3. Présentation du shell :

Le prompt du shell est paramétrable à souhait avec possibilité de coloriage. Tout cela est géré par la variable `PS1` qui comme avec le path, existe en 2 versions : la globale (à définir dans `/etc/profile`) et la locale (à définir dans `~/bashrc` ou `~/bash_profile`), sachant que c'est toujours la locale qui l'emporte. Quelques exemples basiques :

```
export PS1='bonjour ' affiche => bonjour
```

```
export PS1='je_suis_root#' affiche => je_suis_root#
```

```
export PS='C :\>' hum ...
```

Notez l'espace avant la dernière apostrophe qui permet une meilleure visibilité. On peut faire beaucoup mieux en utilisant les variables prédéfinies :

\d pour ajouter la date (format anglais)
 \t pour ajouter l'heure (HH :MM :SS)
 \u pour ajouter le nom de l'utilisateur
 \r pour un retour à la ligne
 \w pour ajouter le chemin complet du répertoire courant
 \W pour ajouter le répertoire courant
 \h pour ajouter le nom de la machine
 \\$ pour afficher une terminaison personnalisée (\$ pour les utilisateurs et # pour root)

Quelques exemples plus évolués :

```

export PS1='bonjour il est \t ' affiche => bonjour il est 17 :30 :51
export PS1='[\u@\h \W]\$ ' affiche chez moi => [kernel@slackware usr]$
  
```

On peut même rajouter de la couleur, pour ça je vous conseille de renommer les variables directement dans votre fichier, ainsi il vous suffira d'utiliser le nom humain pour appeler la bonne couleur. Une couleur finit là où commence l'autre. Voici un exemple de shell très coloré :

```

red='\e[0;31m'
RED='\e[1;31m'
green='\e[0;32m'
GREEN='\e[1;32m'
yellow='\e[0;33m'
YELLOW='\e[1;33m'
blue='\e[0;34m'
BLUE='\e[1;34m'
magenta='\e[0;35m'
MAGENTA='\e[1;35m'
cyan='\e[0;36m'
CYAN='\e[1;36m'
white='\e[0;37m'
WHITE='\e[1;37m'
export PS1="$GREEN[$BLUE\u@\h $YELLOW\W$GREEN]\$"
  
```

Les codes couleurs étant :

```

Noir 0;30
Rouge 0;31
Vert 0;32
Brun 0;33
Bleu 0;34
Violet 0;35
Cyan 0;36
Gris Clair 0;37
Gris 1;30
  
```

```
Rose 1 ;31
Vert Clair 1 ;32
Brun Clair 1 ;33
Bleu Clair 1 ;34
Violet Clair 1 ;35
Cyan Clair 1 ;36
Blanc 1 ;37
```

0.1.6 4. Dircolors : agrémentez votre ls

Si vous rêviez de modifier à votre guise les couleurs affichées par la commande `ls`, cette partie est pour vous ! Pour les autres, c'est bien dommage. Pour commencer, il suffit d'exporter en root le contenu du fichier `/usr/bin/dircolors` dans le répertoire personnel de root (sinon, on pourra l'ajuster par la suite pour le rendre accessible à l'utilisateur de notre choix). La commande consiste en ceci :

```
# dircolors -p > ~/.dircolors
```

(> se chargeant de rediriger le résultat de la commande dans le fichier indiqué qui suit)

Vous pouvez omettre l'argument `-p`, mais ce ne sera pas gagné pour lire aisément le fichier. Une fois que votre fichier est créé dans `/root/.dircolors`, il vous faut l'éditer avec l'éditeur de votre choix, par exemple :

```
# gedit ~/.dircolors &
```

Lorsque vous l'avez sous les yeux, descendez juste en-dessous des codes relatifs aux couleurs (`# 40=black 41=red...`). C'est là que vous aurez à modifier ce qui vous semblera le plus juste. Pour se faire, rien de mieux que de connaître d'avance à quoi correspondent les chiffres !

```
00 : Écriture normale
01 : Texte en gras
02 : Texte en gris pâle
04 : Souligné
07 : Fond noir
08 : Texte en blanc
09 : "overligné" (bref, la ligne comme ceci ^)
[10 à 30 : N'ont rien produits ]
31 : Texte en rouge
32 : Texte en vert
33 : Texte en orange
34 : Texte en bleu
35 : Texte en magenta
36 : Texte en vert turquoise
37 : Texte en gris pâle
38 : Texte souligné
39 : Texte normal
40 : Fond noir
```

```

41 : Fond rouge
42 : Fond vert
43 : Fond orange
44 : Fond bleu
45 : Fond magenta
46 : Fond vert-turquoise
47 : Fond gris pâle

```

Note : Les codes 05 et 08 n'ont pas renvoyé l'effet supposé, mais essayez-les, on ne sait jamais ! Maintenant, les codes n'auront plus de secrets pour vous ! Analysons un petit exemple pour se donner une idée concrète des capacités :

```
* DIR 01 ; 34
```

Si l'on se réfère à la légende ci-dessus, cela signifie que lorsque la commande `ls` sera évoquée, les répertoires seront listés en gras, mais aussi en bleu. Mais on peut faire plus encore ! Reprenons l'exemple en ajoutant une partie à la toute fin :

```
* DIR 01 ; 37 ; 42
```

Les dossiers apparaîtront maintenant en gras, avec un texte gris et un fond vert. Il vous est alors possible de personnaliser au maximum les différentes commandes et, sans oublier, d'ajouter ou enlever les extensions désirées (celles qui n'auront pas été spécialement signifiées suivront l'affichage standard que vous pouvez définir par `NORMAL`, un peu au-dessus de `DIR`). Après avoir établi vos préférences, il serait bien de les enregistrer. La prochaine étape consiste à ajouter à votre fichier `~/.bashrc` (ou `~/.bash_profile`) l'indication suivante :

```
eval `dircolors -b ~/.dircolors`
```

À partir de ce moment, quand vous ouvrirez une console et que vous serez connecté sous `root`, vos jolies couleurs seront là pour donner un peu de vie à la commande `ls` ! Mais il est possible, bien sûr, de rendre ce petit truc accessible aux autres utilisateurs ! Pour cela, copiez en `root` le contenu de votre `/root/.dircolors` ou utilisez l'original destiné aussi à cette fin (`/etc/DIR_COLORS`). Collez-le dans un nouveau fichier que vous appellerez `.dircolors` et enregistrez-le dans le dossier de l'utilisateur visé. Entrez la commande qui suivra pour modifier le propriétaire afin que le fichier appartienne désormais au simple user :

```
# chown nom_du_user :nom_du_user ~/.dircolors
```

Ensuite, dans le fichier `/etc/bashrc`, sous (ou équivalent) :

```
# System wide functions and aliases
# Environment stuff goes in /etc/profile
```

ajoutez :

```
eval `dircolors -b ~/.dircolors`
```

De cette manière, tous les utilisateurs disposant d'un fichier `.dircolors` leur appartenant dans leur répertoire personnel pourront aller personnaliser les couleurs de leur propre commande `ls` facilement sans devoir demander à l'administrateur de le faire à leur place ! Comme le fichier `/etc/bashrc` fut au bout du compte édité, vous pouvez retirer de `/root/.bashrc` la commande `eval `dircolors -b ~/.dircolors`` puisqu'elle est déjà présente ailleurs. J'espère que cette astuce vous poussera davantage à utiliser cette fameuse `ls` ! À bientôt pour de nouvelles découvertes encore plus alléchantes !