Reprenez le contrôle à l'aide de Linux !

2 mois Facile Linux, OS

NANO, L'ÉDITEUR DE TEXTE DU DÉBUTANT

Nousavons découvert plusieurs façons de voir le contenu d'un fichier en console.
Mais… aucune des commandes que nous avons étudiées ne nous permettait d'éditer un fichier.

Pourquoi ai-je repoussé le moment où je vous parlerais des éditeurs de texte ? Parce que c'est un des domaines les plus riches de la console ! Parmi les plus célèbres éditeurs de texte console de Linux, il faut connaître : Nano, Vim et Emacs.

Que de jolis noms, n'est-ce pas ?
Des trois que je viens de citer, **Nano** est de loin le plus simple à utiliser. Ce n'est pas pour rien que ce chapitre s'intitule « Nano, l'éditeur de texte du débutant ».

Nous découvrirons Vim plus loin dans ce livre, car il est plus complexe et nécessite déjà un bon niveau.

Premiers pas avec Nano

En sciences, le terme « nano » représente une toute petite unité. Par exemple, un atome a une taille d'environ 0,1 nanomètre.

Si l'éditeur de texte que je vais vous présenter s'appelle Nano, c'est parce qu'il est tout petit. Il s'agit d'un programme très simple comparé à Vim et Emacs et il nous conviendra tout à fait pour démarrer. Il possède assez peu de fonctions par rapport aux deux autres logiciels (qui peuvent devenir de véritables machines de guerre) mais suffisamment pour commencer à vous débrouiller avec un éditeur de texte.

Nano est un éditeur de texte, pas un traitement de texte !

Savez-vous vraiment ce qu'est un éditeur de texte ? Ne le confondez-vous pas avec un traitement de texte ?

Un **éditeur de texte** est un programme qui permet de modifier des fichiers de texte brut, sans mise en forme (gras, italique, souligné…). Sous Windows, on dispose d'un éditeur de texte très basique : le Bloc-Notes. Sous Linux, on a le choix entre Nano, Vim, Emacs et bien d'autres, sachant qu'au moins un de ceux-là est installé par défaut sur la plupart des distributions.

Un **traitement de texte** est fait pour rédiger des documents mis en forme. Sous Windows, Word est le plus célèbre traitement de texte ; sous Linux, on possède l'équivalent : Open Office Writer. Ces programmes ne peuvent être utilisés qu'en mode graphique, la console ne permettant pas vraiment de faire de la mise en forme.

> *Quand a-t-on besoin d'un éditeur de texte ?*

Chaque fois que vous devez éditer un fichier de texte brut. Sous Windows, vous avez l'habitude de voir des fichiers de texte brut au format `.txt`. Sous Linux, vous savez que l'extension importe peu (on peut trouver des fichiers en texte brut sans extension).

Les éditeurs de texte sont parfaits pour les programmeurs en particulier : ils permettent d'éditer des fichiers `.c`, `.cpp`, `.h`, `.rb`, `.py`, etc. (En fonction de votre langage de programmation.)
Même si vous ne programmez pas, vous aurez besoin d'utiliser un éditeur de texte pour modifier des fichiers de configuration. Ces fichiers n'ont pas d'extension particulière, mais à force vous apprendrez à les reconnaître.

Après avoir appris à utiliser Nano, nous nous ferons les dents sur nos premiers fichiers de configuration : le `nanorc` et le `bashrc`. Ce sera l'occasion pour vous de personnaliser votre Nano et votre console.;-)

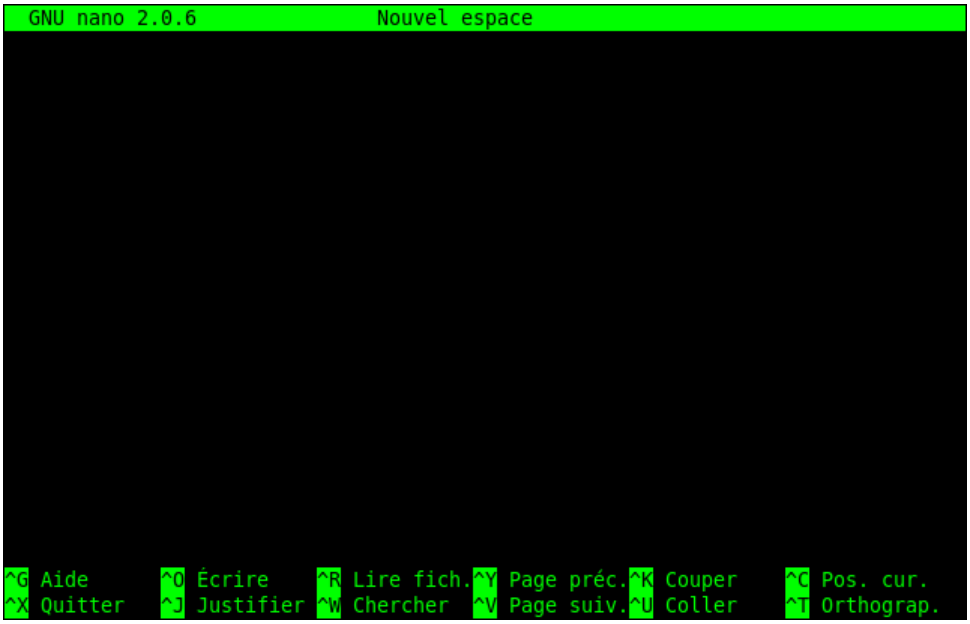
Découverte de Nano

Le nom complet de Nano est « GNU nano », en référence au projet GNU dont je vous ai parlé dans le tout premier chapitre. Il s'agit d'un logiciel qui s'inspire de « pico », un éditeur de texte plus ancien qui se voulait lui aussi très simple d'utilisation.

Pour démarrer le logiciel, il vous suffit simplement de taper `nano` dans la console :

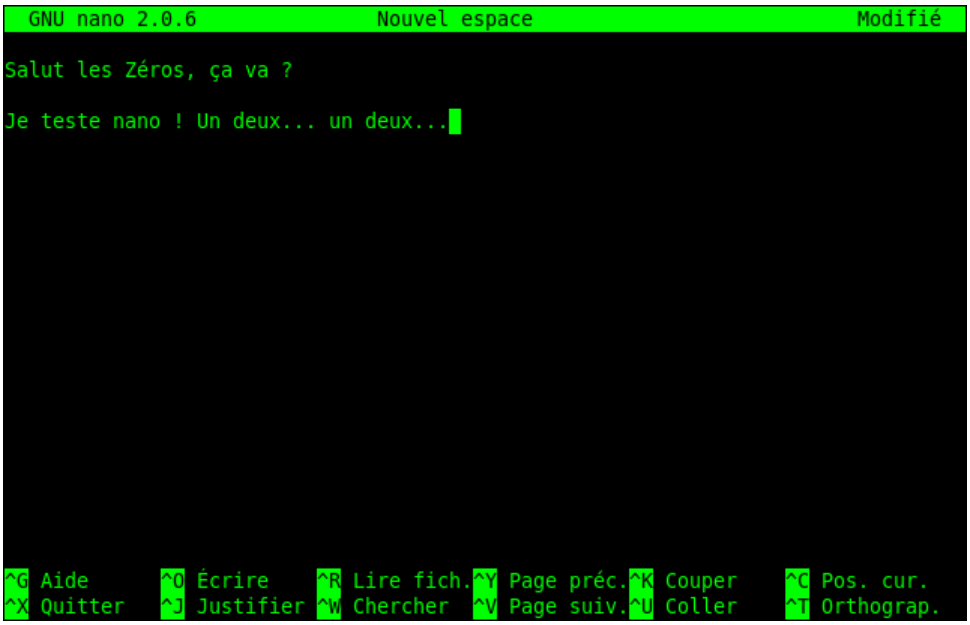
console
```
nano
```

L'éditeur Nano s'ouvre immédiatement (figure suivante).



Nano

Dès lors, vous pouvez commencer à taper du texte (exemple sur la figure suivante).



Nano : écriture de texte

C'est aussi simple que cela !
Ne riez pas, je précise qu'il « suffit de taper du texte » car ce n'est pas aussi simple sous d'autres éditeurs, comme Vim par exemple.

Les raccourcis clavier de Nano

En bas de votre écran, vous pouvez voir un espace d'aide (figure suivante). Que signifie-t-il exactement ?
Il s'agit d'un aide-mémoire pour vous rappeler à tout moment les commandes principales que vous pouvez lancer sous Nano.

Aide de Nano

Le symbole ^ signifie `Ctrl` (la touche `Contrôle` de votre clavier). Ainsi, pour quitter Nano, il suffit de taper `Ctrl + X`.

Voici les raccourcis les plus importants :

- `Ctrl + G` : afficher l'aide ;
- `Ctrl + K` : couper la ligne de texte (et la mettre dans le presse-papier) ;
- `Ctrl + U` : coller la ligne de texte que vous venez de couper ;
- `Ctrl + C` : afficher à quel endroit du fichier votre curseur est positionné (numéro de ligne…) ;

- `Ctrl + W` : rechercher dans le fichier ;
- `Ctrl + O` : enregistrer le fichier (écrire) ;
- `Ctrl + X` : quitter Nano.

Vous pouvez vous déplacer dans le fichier avec les flèches du clavier ainsi qu'avec les touches `Page Up` et `Page Down` pour avancer de page en page (les raccourcis `Ctrl + Y` et `Ctrl + V` fonctionnent aussi).

Si l'aide-mémoire vous encombre, vous pouvez gagner de la place en appuyant sur `Échap` puis sur `X`. Vous pouvez l'afficher de nouveau avec la même suite de touches.

La recherche

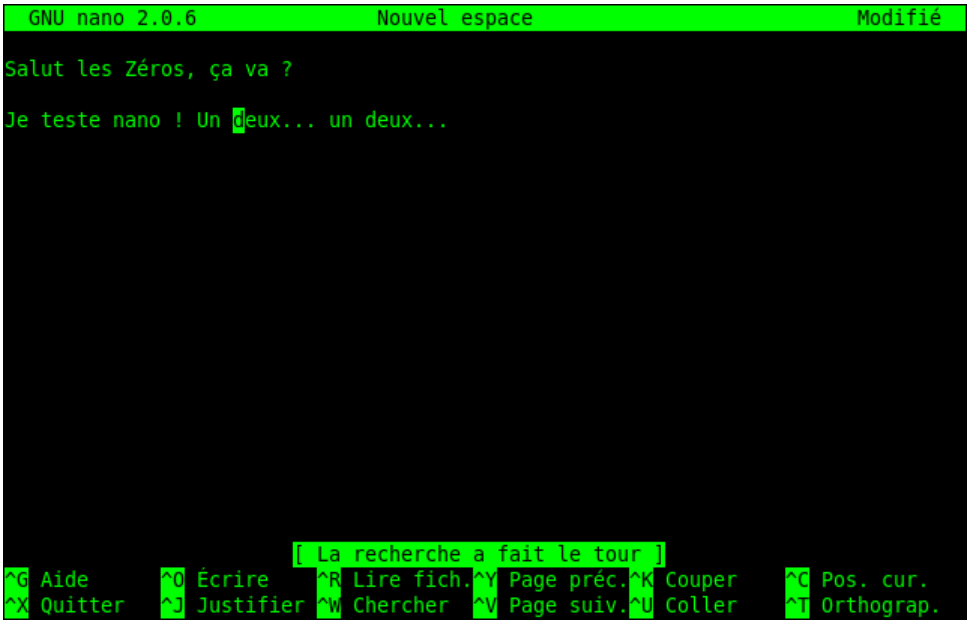
La combinaison de touches `Ctrl + W` lance une recherche dans le fichier (figure suivante).

Recherche dans Nano

Il vous suffit d'écrire le mot que vous recherchez (figure suivante)…

Recherche dans Nano

… puis de taper `Entrée` (figure suivante).



Recherche dans Nano

Le curseur est automatiquement positionné à la première occurrence trouvée. Si le curseur est à la fin, la recherche recommence du début.

> Si vous voulez sortir du mode recherche, tapez `Ctrl + C` (Annuler).

Si vous souhaitez aller au résultat suivant (au « deux » suivant), faites à nouveau `Ctrl + W` pour lancer une recherche. La recherche précédente est sauvegardée et apparaît entre crochets. Si vous voulez rechercher le même mot (et donc aller au résultat suivant), tapez juste `Entrée` sans écrire de mot à rechercher (figure suivante).

Recherche dans Nano

Enregistrer et quitter

Pour enregistrer à tout moment, faites `Ctrl + O`.
Si vous essayez de quitter (`Ctrl + X`) sans enregistrer auparavant, un message vous demandera si vous voulez sauvegarder (figure suivante).

Confirmation de sortie de Nano

Si vous appuyez sur la touche `o`, vous passerez en mode enregistrement.
Si vous appuyez sur la touche `n`, Nano quittera sans enregistrer.
Si vous utilisez la combinaison `Ctrl + C`, vous annulerez votre demande de sortie de Nano et ne quitterez donc pas le logiciel.

En appuyant sur `o`, vous vous retrouvez en mode enregistrement. Tapez juste le nom du fichier que vous voulez créer puis pressez `Entrée` (figure suivante).

Enregistrement dans Nano

Après ça, Nano sera fermé et vous retrouverez votre bonne vieille ligne de commandes.

Les paramètres de la commande Nano

Lorsque vous appelez Nano dans la ligne de commandes, vous pouvez spécifier plusieurs paramètres. Le plus courant est d'indiquer en paramètre le nom du fichier qu'on veut ouvrir. Ainsi :

```
console
nano salut.txt
```

... ouvrira le fichier `salut.txt` que l'on vient de créer.

Si le fichier n'existe pas, il sera automatiquement créé par Nano lors du premier enregistrement.

À part ça, la commande `nano` accepte de nombreux paramètres. Pour vous, j'en ai sélectionné trois qui me semblent faire partie des plus utiles.

- **-m** : autorise l'utilisation de la souris sous Nano. En console, oui, oui. Vous pouvez vous en servir pour cliquer avec votre souris sur la zone de texte où vous voulez placer votre curseur.
- **-i** : indentation automatique. L'alinéa (tabulations) de la ligne précédente sera respecté lorsque vous irez à la ligne. Très utile lorsque vous éditez

un fichier de code source.

- **-A** : active le retour intelligent au début de la ligne. Normalement, lorsque vous appuyez sur la touche `Origine` (aussi connue sous le nom de `Home`) située à côté de la touche `Fin`, le curseur se repositionne au tout début de la ligne. Avec cette commande, il se positionnera après les alinéas. Comme `-i`, il s'agit d'une option utile avant tout pour les programmeurs.

Si je veux lancer Nano avec toutes ces options à la fois, je peux donc écrire :

console
```
nano -miA salut.txt
```

Configurer Nano avec .nanorc

Vous savez maintenant utiliser Nano. Comme vous avez pu le voir, ce n'est pas très compliqué. Il suffit d'apprendre un peu les raccourcis clavier les plus utiles et on peut rapidement s'en servir.

Justement… et si on utilisait Nano pour quelque chose d'utile ? Non parce que bon, le fichier `salut.txt` est sympa, mais ça ne va pas nous faire avancer.

Alors pour l'occasion, je me suis dit que j'allais vous faire éditer quelques fichiers de configuration. Par exemple, il existe un fichier de configuration de Nano qui indique toutes vos préférences. Celui-ci s'appelle `.nanorc`.

Pourquoi `.nanorc` ?

La plupart des fichiers de configuration commencent par un point. Cela permet de « cacher » le fichier quand on fait un `ls`. Bien entendu, comme vous devriez maintenant le savoir, les fichiers cachés peuvent toujours être affichés en utilisant le paramètre `-a` : `ls -a`.

Chaque utilisateur de la machine peut créer son propre fichier de configuration `.nanorc` dans son répertoire personnel (home). Chez moi, ce fichier doit être placé à la position : `/home/mateo21/.nanorc`. Ce fichier est lu par Nano à chaque fois que vous le démarrez.

> *Je viens de regarder la liste des fichiers de mon home, mais même en incluant les fichiers cachés avec `-a` je ne vois pas de fichier appelé `.nanorc` !*

En effet, il se peut que le fichier `.nanorc` n'existe pas chez vous. Si tel est le cas, Nano sera chargé avec les options par défaut.

Création du `.nanorc`

Pas de `.nanorc` ? Pas de problème, il suffit de le créer. On peut par exemple faire ceci :

console
```
nano .nanorc
```

Cette commande ouvre Nano. Comme le fichier `.nanorc` n'existe pas, un document vide est ouvert (figure suivante). Le fichier `.nanorc` sera créé lorsque vous enregistrerez.

Dans ce fichier, vous devez écrire une commande par ligne.
Chaque commande commence par un **set** (pour activer) ou un **unset** (pour désactiver) suivi de l'option qui vous intéresse.

Par exemple, pour activer la souris, écrivez :

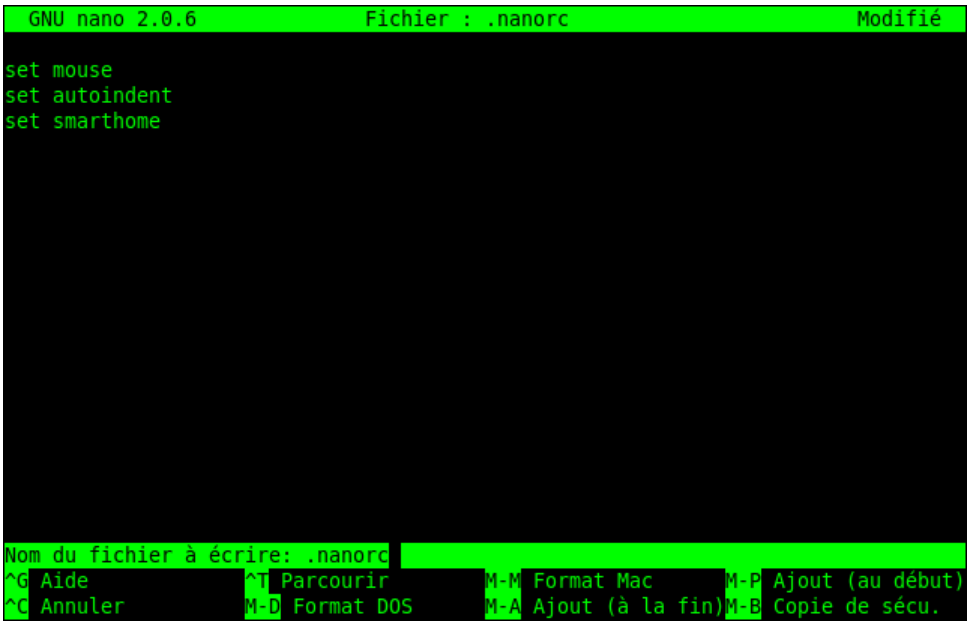
console
```
set mouse
```

Ainsi Nano sera automatiquement chargé avec la prise en charge de la souris. Vous n'aurez pas besoin de réécrire systématiquement le paramètre `-m` qu'on a vu tout à l'heure.

On peut faire de même pour éviter d'avoir à taper à chaque fois les paramètres `-i` et `-A` avec d'autres séries de `set`. Au final, on écrira ceci :

console
```
set mouse
set autoindent
set smarthome
```

Enregistrez le fichier avec `Ctrl + O`. Comme vous avez déjà mentionné le nom du fichier en paramètre lors de l'ouverture de Nano, celui-ci sera automatiquement écrit pour vous (figure suivante).



Vous pouvez ensuite faire `Ctrl + X` pour quitter Nano.
Je vous rappelle que pour que ces options soient prises en compte, il faut démarrer une nouvelle session de Nano (c'est pour ça que la souris n'a pas automatiquement fonctionné dès que vous avez enregistré le fichier).

Si vous relancez Nano ensuite, vous pouvez constater que la souris fonctionne et que les options d'indentation automatique et de retour à la ligne intelligent sont elles aussi opérationnelles. :-)

Le `nanorc` global et la coloration syntaxique

Ce fichier `.nanorc` dans votre home est très pratique car il vous permet de définir vos propres options. Mais si vous avez dix utilisateurs sur votre machine et que vous voulez activer le support de la souris pour tout le monde, vous n'allez quand même pas créer un fichier `.nanorc` pour chacun !

Il existe un fichier `nanorc` « global » qui est pris en compte pour tout le monde. Celui-ci est situé dans `/etc/nanorc` (attention : il n'y a pas de point devant, cette fois.)
Ce fichier ne peut être modifié que par root. Je vous conseille donc de l'ouvrir avec un `sudo` (ou dans une console en root si vous avez fait `sudo su` avant) :

console
```
sudo nano /etc/nanorc
```

Normalement, ce fichier existe déjà. Comme vous pouvez le constater sur la figure suivante, il est bien rempli.

Il sert en fait de fichier d'exemple. Toutes les options disponibles dans un `.nanorc` sont présentes, mais elles sont précédées d'un `#` qui signifie qu'il s'agit d'un commentaire. Les commentaires sont ignorés par Nano.

Le début du fichier vous explique (en anglais) que c'est un fichier d'initialisation d'exemple de Nano.

Après le petit blabla d'introduction, vous avez la liste des options disponibles. Toutes sont commentées.
La première est `autoindent`.

console
```
# set autoindent
```

Supprimez juste le `#` pour décommenter la ligne et donc pour activer l'indentation automatique pour tous les utilisateurs.

console
```
set autoindent
```

Vous pouvez parcourir le fichier à la recherche d'options intéressantes que vous voulez activer.

Vers la fin, vous verrez une section appelée « color setup », qui commence par ces lignes-là :

console
```
## Nanorc files
# include "/usr/share/nano/nanorc.nanorc"

## C/C++
# include "/usr/share/nano/c.nanorc"

## HTML
# include "/usr/share/nano/html.nanorc"
```

Je vous invite à décommenter toutes les lignes d'`include`. Cela permettra d'activer la coloration « intelligente » de vos fichiers selon leur type. Vous pourrez ainsi avoir des fichiers HTML colorés, des fichiers C colorés, des fichiers `nanorc` colorés, etc.

Enregistrez le fichier puis quittez Nano.

> Si vous avez une erreur lors de l'enregistrement, cela signifie que vous n'avez pas ouvert le fichier en root. Seul root a le droit de modifier ce fichier. Fermez Nano et relancez-le avec un `sudo` cette fois.

Configurer sa console avec .bashrc

Tout comme il existe un fichier de configuration de Nano, il existe un fichier de configuration de l'ensemble de la console : le `.bashrc`. Il se situe dans votre répertoire personnel et celui-ci existe déjà normalement.

console
```
mateo21@mateo21-desktop:/usr/share/nano$ cd
mateo21@mateo21-desktop:~$ nano .bashrc
```

Édition du `.bashrc` personnel

> Le fichier `.bashrc` est un peu complexe pour les simples mortels que nous sommes (pour le moment), donc attention à ne pas éditer n'importe quoi au risque de tout casser.
> Bref, soyez juste un peu attentifs et tout ira bien.

Nous n'allons pas nous intéresser au `.bashrc` en détail. Nous allons seulement voir quelques lignes faciles à éditer qui vous permettront de personnaliser un peu votre console.

Personnaliser l'invite de commandes

Le fichier `.bashrc` vous permet entre autres choses de personnaliser l'invite de commandes. Vous savez, ce petit message qui s'affiche devant votre curseur dans la console :

console
```
mateo21@mateo21-desktop:~$
```

Rendez-vous plus bas dans le fichier, jusqu'à ce que vous tombiez sur ces lignes :

console
```
# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
xterm-color)
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
    ;;
*)
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
    ;;
esac

# Comment in the above and uncomment this below for a color prompt
# PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
```

Dans les commentaires, on vous dit que vous pouvez activer l'invite de commandes colorée en commentant les lignes du dessus et en décommentant la dernière ligne.
Rajoutez donc un `#` devant les deux premiers `PS1`, et enlevez le `#` devant le dernier `PS1` pour que la coloration de l'invite de commandes puisse fonctionner :

console
```
# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
```

```
xterm-color)
#    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
   ;;
*)
#    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
   ;;
esac

# Comment in the above and uncomment this below for a color prompt
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
```

Enregistrez. Il faudra ouvrir une nouvelle console pour que la modification soit prise en compte afin de profiter d'une invite de commandes en couleurs.

Si vous êtes en forme, vous pouvez éditer la ligne que vous venez de décommenter :

console

```
PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]: \[\033[01;34m\]\w\[\033[00m\]\$ '
```

C'est en fait elle qui indique ce que l'invite de commandes doit afficher. Les séquences de type « `\033` » servent à paramétrer la couleur (ce n'est pas simple, je vous l'accorde).
Le symbole `\u` au milieu indique le nom de l'utilisateur (mateo21 par exemple) et `\h` indique le nom de la machine hôte (mateo21-desktop). Vous pouvez repérer dans cette ligne le « @ » qui sépare les deux, le « : », le « $ », etc.

Vous pouvez essayer de personnaliser un peu ces éléments ainsi que leur ordre si ça vous amuse (mais faites quand même attention à ne pas mettre le bazar là-dedans, hein. ;-).

Créer des alias

Les alias sont des commandes que vous créez et qui sont automatiquement transformées en d'autres commandes.
Descendez un peu plus bas dans le fichier, vous trouverez des lignes commentées commençant par « alias ».

Je vous invite à les personnaliser comme moi pour commencer :

console

```
# enable color support of ls and also add handy aliases
if [ "$TERM" != "dumb" ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=auto'
    #alias dir='ls --color=auto --format=vertical'
    #alias vdir='ls --color=auto --format=long'
fi


# some more ls aliases
#alias ll='ls -l'
#alias la='ls -A'
#alias l='ls -CF'
```

Vous avez déjà probablement un alias créé :

console

```
alias ls='ls --color=auto'
```

Celui-ci active la coloration des résultats d'un `ls` à chaque fois que vous tapez `ls`. En fait, `ls` est systématiquement et automatiquement transformé par la console en `ls --color=auto`. C'est quand même plus rapide que de réécrire sans cesse ces paramètres.

Il y a un autre alias que j'ai l'habitude d'utiliser, c'est `ll` (deux fois la lettre L minuscule). Cela permet de faire un `ls` en mode détaillé.
Personnellement, j'ai un peu complété l'alias pour utiliser plus d'options à la fois, comme j'en ai parlé dans le chapitre sur `ls` :

console

```
alias ll='ls -lArth'
```

… signifie que la commande `ll` fera appel à `ls` avec les options qui permettent d'afficher le détail de chaque fichier, d'afficher les fichiers cachés, d'afficher les fichiers dans l'ordre inverse de dernière modification (le fichier le plus récent sera en bas) et d'afficher des tailles de fichiers lisibles pour un humain (`-h`).

La commande `ls` appellera automatiquement l'alias `ls --color=auto`, ce qui fait qu'un `ll` sera aussi coloré. Bref, c'est un peu un alias en chaîne.

> Si vous tapez la commande `alias` dans la console, vous verrez la liste de tous les alias définis pour votre utilisateur.

Vous pouvez vous aussi définir vos propres alias. Comme vous pouvez le voir, c'est très simple car cela fonctionne sur le modèle :

console

```
alias nom='commande'
```

Attention à ne pas mettre d'espace autour du symbole « = ».

On peut par exemple en profiter pour sécuriser un peu nos `rm` pour éviter que l'on puisse supprimer tout le système depuis la racine `/`. Il y a en effet un paramètre de sécurité disponible avec `rm` : `--preserve-root`. Mais ce serait un peu long de l'écrire à chaque fois et on risquerait surtout d'oublier. En définissant un alias sur `rm`, vous ne pourrez pas oublier :

console

```
alias rm='rm --preserve-root'
```

> Ne testez pas l'efficacité de cette commande en faisant un `rm -rf /` en root ! En effet, il faut relancer une console pour que les modifications soient prises en compte, et si vous avez fait une faute de frappe dans votre alias, vous ne serez pas protégés… mais pendant ce temps votre système sera détruit, lui !
> Bref, même pour « vérifier », ne vous amusez pas à utiliser le `rm` de la mort…

Édition du `bashrc` global

Si vous voulez définir des alias ou modifier l'invite de commandes pour tous vos utilisateurs, vous pouvez le faire en une seule fois en éditant le fichier `bashrc` global situé dans : `/etc/bash.bashrc`.
Ce `bashrc` doit être édité en root.

Ce fichier propose un peu moins d'exemples commentés que celui présent dans votre home. Vous pouvez y copier vos alias et la ligne définissant l'invite de commandes (commençant par `PS1`).

> Les éléments du `bashrc` personnel ont la priorité sur ceux du `bashrc` global. Si un même alias est défini dans les deux, c'est celui du `bashrc` personnel qui sera pris en compte.

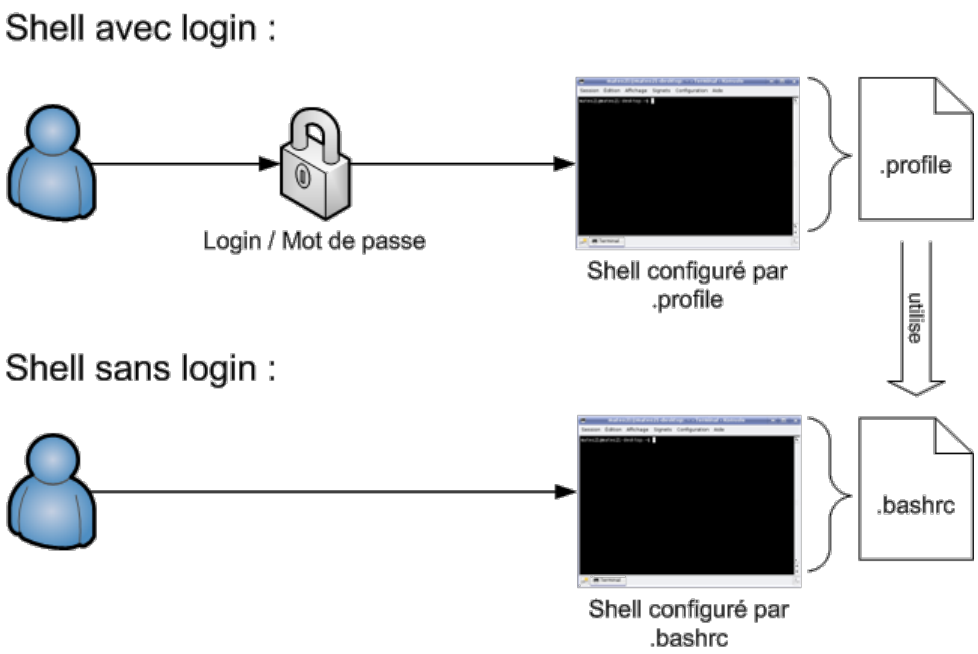
Et aussi... le `.profile`

De même qu'il existe un ~/.bashrc et un `/etc/bash.bashrc`, il existe un ~/.profile et un `/etc/profile`. Quelle est la différence ?

En gros, le **.profile** est lu à chaque nouvelle console dans laquelle vous vous loggez (vous rentrez votre login / mot de passe). C'est le cas des consoles que vous ouvrez avec `Ctrl + Alt + F1` à `F6` (`tty1` à `tty6`).
Le **.bashrc** est lu lorsque vous ouvrez une console dans laquelle vous ne vous loggez pas. C'est le cas des consoles que vous ouvrez en mode graphique (Terminal sous Unity, Konsole sous KDE).

C'est un peu compliqué pour pas grand-chose au final. Dans la pratique, le `.profile` fait par défaut appel au `.bashrc`... Donc il suffit d'éditer votre `.bashrc` et vous modifierez ainsi les options de toutes vos consoles : celles avec et sans login. Voilà pourquoi je vous ai fait éditer dès le début le `.bashrc`. ;-)

Pour bien comprendre comment ça fonctionne, tout est résumé dans le schéma de la figure suivante.



Le fichier .profile appelle le .bashrc

> Le shell est le programme qui interprète les commandes que vous tapez (vous pouvez considérer que c'est un synonyme de « console »).

On a, sur ce schéma, deux types de shell possibles :

- soit on a ouvert un shell qui demande un mot de passe et dans ce cas, c'est le `.profile` qui est lu pour la configuration ;
- soit on a ouvert un shell qui ne demande pas de mot de passe (c'est le cas d'une console en mode graphique en général) et dans ce cas-là, c'est le `.bashrc` qui servira à la configuration.

La particularité, comme le montre le schéma, c'est que le `.profile` fait appel au `.bashrc`... ce qui signifie que vous pouvez faire toutes vos configurations dans le `.bashrc` pour qu'elles soient valables quel que soit le type de shell que vous ouvrez.

En résumé

- Un éditeur de texte est un programme qui ouvre des fichiers texte (un peu comme Bloc-Notes sous Windows). On en a régulièrement besoin sous Linux pour modifier des fichiers de configuration, par exemple.
- Il existe de nombreux éditeurs de texte en console qui peuvent être très complets, comme Vim et Emacs.
- L'éditeur Nano est un des éditeurs en console les plus simples à utiliser ; nous commençons donc par découvrir celui-ci.

- On utilise plusieurs raccourcis clavier dans un éditeur de texte comme Nano. `Ctrl + W` lance une recherche, `Ctrl + O` enregistre le fichier, `Ctrl + X` permet de quitter, etc.
- On peut utiliser Nano pour modifier son fichier de configuration `.bashrc` et personnaliser sa console. On peut notamment s'en servir pour colorer l'invite de commandes et créer des alias.

Les utilisateurs et les droits

Installer des programmes avec apt-get

Manipuler la console et les fichiers

1. La console, ça se mange ?
2. Entrer une commande
3. La structure des dossiers et fichiers
4. Manipuler les fichiers
5. Les utilisateurs et les droits
6. **Nano, l'éditeur de texte du débutant**
7. Installer des programmes avec apt-get
8. RTFM : lisez le manuel !
9. Rechercher des fichiers

L'auteur

Mathieu Nebra

Entrepreneur à temps plein, auteur à temps plein et fondateur d'OpenClassrooms :o)

Découvrez aussi ce cours en...

Premium
eBook

Livre papier

PDF