Base

Astier Guillaume

05/01/2026

Système de fichiers

man - RTFM

commande

Commandes de base

Chaînage de commandes

Edition sous LINUX/UNIX en mode TEXTUEL

Les Bases de vim

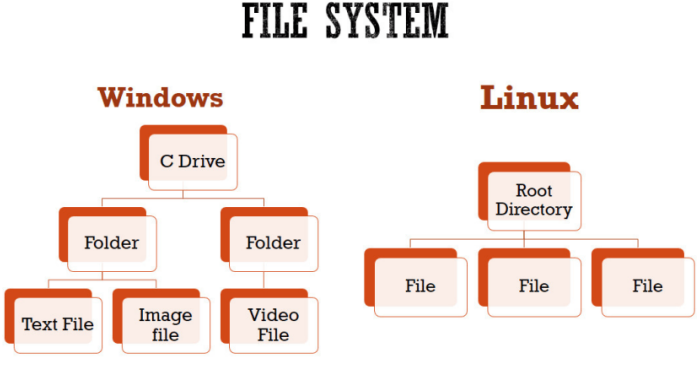
Protéger une variable

Système de fichiers

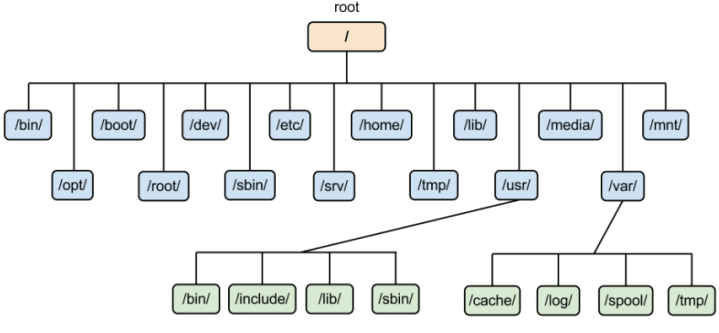
Système de fichiers

Le *F*ileSystem *H*ierarchy *S*tandard (FHS) définit la structure arborescente et le contenu des principaux répertoires des systèmes de fichiers des systèmes d'exploitation GNU / Linux et de la plupart des systèmes Unix. Nous en sommes à la version 3 depuis 2015.

Windows


FILE SYSTEM
Windows
C Drive
Folder
Folder
Text File
Image file
Video File
Linux
Root Directory
File
File
File

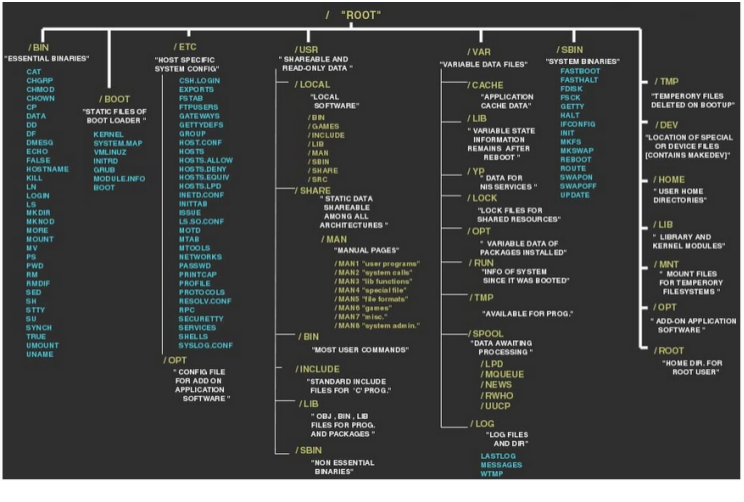
Linux


root
/
/bin/
/boot/
/dev/
/etc/
/home/
/lib/
/media/
/mnt/
/opt/
/root/
/sbin/
/srv/
/tmp/
/usr/
/var/
/bin/
/include/
/lib/
/sbin/
/cache/
/log/
/spool/
/tmp/

Détail de l'arborescence Linux

Répertoire	Contenu
/bin	Binaires (exécutables = commandes essentielles)
/boot	Fichiers statiques pour le chargeur d'amorçage
/dev	Fichiers de pilotes de périphériques
/etc	Fichiers de configuration système
/home	Répertoires personnels des utilisateurs
/lib	Bibliothèques partagées et modules essentiels du noyau
/media ou /mnt	Points de montage pour les médias amovibles
/proc	Répertoire virtuel pour les informations système
/root	Répertoire personnel de l'utilisateur root
/sbin	Exécutables système essentiels
/tmp	Fichiers temporaires
/usr	Hiérarchie secondaire
/var	Données variables
/opt	Répertoire pour d'autres logiciels

Vue de l'arborescence Linux



man - RTFM

man - RTFM

La commande **man** est un manuel intégré pour l'utilisation des commandes Linux. Elle permet aux utilisateurs d'afficher les manuels de référence d'une commande ou d'un utilitaire exécuté dans le terminal.

La *page man* (abréviation de *manual page*) contient une description de la commande, ses options, ses drapeaux (flags), des exemples et d'autres sections informatives.

commande

commande

La grande majorité des commandes Linux disposent d'une page man.

Donc, si tu veux savoir comment utiliser une commande, il te suffit de la préfixer par **man** :

```
username@hostname:/tmp$ man ls
LS(1) User Commands LS(1)

NOM
ls - liste le contenu 'dun répertoire

SYNOPSIS
ls [OPTION]... [FICHIER]...

DESCRIPTION
Liste des informations sur les FICHIERS (le répertoire courant par défaut).
Trie les entrées par ordre alphabétique si aucune des options -cftuvSUX ni --sort
   'nest spécifiée.

   Les arguments obligatoires pour les options longues sont également obligatoires
       pour les options courtes.

   -a, --all
          ne pas ignorer les entrées commençant par .

   -A, --almost-all
          ne pas lister . et .. implicites

   --author
          avec -l, afficher 'lauteur de chaque fichier
```

Argument / option

Quand tu tapes des commandes dans un terminal, tu dois entrer des arguments ou des options.

```
username@hostname:/tmp$ ls -l /tmp
```

Ici nous avons :

- `-l` : c'est une option

En réalité, ce sont tous les deux des arguments, mais la commande comprend que `-l` est une option tandis que `/tmp/` est un argument classique.

Argument / option

Quand tu tapes des commandes dans un terminal, tu dois entrer des arguments ou des options.

```
username@hostname:/tmp$ ls -l /tmp
```

Ici nous avons :

- `-l` : c'est une option
- `/tmp/` : c'est un argument

En réalité, ce sont tous les deux des arguments, mais la commande comprend que `-l` est une option tandis que `/tmp/` est un argument classique.

–help

La grande majorité des commandes Linux possèdent une option `-h` ou `--help`.

Cette option (qui est en réalité un argument) te permet d'afficher un résumé simple des options de la commande.

```
username@hostname:/tmp$ ls --help
Utilisation : ls [OPTION]... [FICHIER]...
Afficher des renseignements sur les FICHIERS (du répertoire actuel par défaut).
Trier les entrées alphabétiquement si aucune des options -cftuvSUX ou --sort
ne sont utilisées.
```

Les arguments obligatoires pour les options longues le sont aussi pour les options courtes.

- -a, –all ne pas ignorer les entrées débutant par .

Les arguments obligatoires pour les options longues le sont aussi pour les options courtes.

- -a, –all ne pas ignorer les entrées débutant par .
- -A, –almost-all ne pas inclure . ou .. dans la liste

Les arguments obligatoires pour les options longues le sont aussi pour les options courtes.

- -a, –all ne pas ignorer les entrées débutant par .
- -A, –almost-all ne pas inclure . ou .. dans la liste
- –author avec -l, afficher l'auteur de chaque fichier

-h

username@hostname:/tmp$ useradd -h Utilisation : useradd [options] LOGIN useradd -D useradd -D [options]

Options : –badnames ne pas vérifier les noms incorrects -b, –base-dir REP_BASE répertoire de base pour le répertoire personnel du nouvel utilisateur –btrfs-subvolume-home utiliser un sous-volume BTRFS pour le répertoire home -c, –comment COMMENTAIRE définir le champ « GECOS » du compte [...]

Commandes de base

La commande **RTFM**

La commande la plus puissante sous linux reste le `man`.

Un raccourci pour le manuel d'utilisation.

Vous ne comprenez pas une commande utilisez *man* !!!!!!!!!

ex : Je ne sais pas comment lister le contenu d'un répertoire avec des fichiers cachés :

```
man ls
```

ex : je recherche une commande concernant un sujet, utilisez -k pour le mot clé

```
man -k mysubject
```

Les commandes usuelles (1/3)

- `ls` “LiSt” ==> Liste les fichiers d’un répertoire

Les commandes usuelles (1/3)

- `ls` “LiSt” ==> Liste les fichiers d'un répertoire
- `mkdir` “MaKe DIRectory” ==> Créer un répertoire

Les commandes usuelles (1/3)

- `ls` “LiSt” ==> Liste les fichiers d’un répertoire
- `mkdir` “MaKe DIRectory” ==> Créer un répertoire
- `rmdir` “ReMove DIRectory” ==> Supprimer un répertoire vide

Les commandes usuelles (1/3)

- `ls` “LiSt” ==> Liste les fichiers d’un répertoire
- `mkdir` “MaKe DIRectory” ==> Créer un répertoire
- `rmdir` “ReMove DIRectory” ==> Supprimer un répertoire vide
- `cat` “conCATenate” ==> Affiche le contenu d’un fichier ou de plusieurs fichiers concaténés

Les commandes usuelles (2/3)

- `cp` “CoPy” ==> Copier un fichier

Les commandes usuelles (2/3)

- `cp` “CoPy” ==> Copier un fichier
- `mv` “MoVe” ==> Déplacer ou renommer des fichiers

Les commandes usuelles (2/3)

- `cp` “CoPy” ==> Copier un fichier
- `mv` “MoVe” ==> Déplacer ou renommer des fichiers
- `rm` “ReMove” ==> Supprimer un fichier

Les commandes usuelles (2/3)

- `cp` “CoPy” ==> Copier un fichier
- `mv` “MoVe” ==> Déplacer ou renommer des fichiers
- `rm` “ReMove” ==> Supprimer un fichier
- `ln` “LiNk” ==> Créer des liens

Les commandes usuelles (3/3)

- `umask` “MASQUE Utilisateur” ==> Définir le masque de création de fichier

Les commandes usuelles (3/3)

- `umask` “MASQUE Utilisateur” ==> Définir le masque de création de fichier
- `cut` “CUT” ==> Permet de récupérer un champ spécifique dans une chaine de caractères

Les commandes usuelles (3/3)

- `umask` “MASQUE Utilisateur” ==> Définir le masque de création de fichier
- `cut` “CUT” ==> Permet de récupérer un champ spécifique dans une chaine de caractères
- `file` “FILE” ==> Permet de connaître le type d’un fichier (il n’y a pas d’extension sous linux)

Les commandes usuelles (3/3)

- `umask` “MASQUE Utilisateur” ==> Définir le masque de création de fichier
- `cut` “CUT” ==> Permet de récupérer un champ spécifique dans une chaine de caractères
- `file` “FILE” ==> Permet de connaître le type d’un fichier (il n’y a pas d’extension sous linux)
- `head` ==> Imprimer le début d’un fichier

Les commandes usuelles (3/3)

- `umask` “MASQUE Utilisateur” ==> Définir le masque de création de fichier
- `cut` “CUT” ==> Permet de récupérer un champ spécifique dans une chaine de caractères
- `file` “FILE” ==> Permet de connaître le type d’un fichier (il n’y a pas d’extension sous linux)
- `head` ==> Imprimer le début d’un fichier
- `tail` ==> Imprimer la fin d’un fichier

Les commandes usuelles (3/3)

- `umask` “MASQUE Utilisateur” ==> Définir le masque de création de fichier
- `cut` “CUT” ==> Permet de récupérer un champ spécifique dans une chaine de caractères
- `file` “FILE” ==> Permet de connaître le type d’un fichier (il n’y a pas d’extension sous linux)
- `head` ==> Imprimer le début d’un fichier
- `tail` ==> Imprimer la fin d’un fichier
- `tee` ==> Redirigez le flux standard vers le terminal et vers un fichier

Les commandes usuelles (3/3)

- `umask` “MASQUE Utilisateur” ==> Définir le masque de création de fichier
- `cut` “CUT” ==> Permet de récupérer un champ spécifique dans une chaine de caractères
- `file` “FILE” ==> Permet de connaître le type d’un fichier (il n’y a pas d’extension sous linux)
- `head` ==> Imprimer le début d’un fichier
- `tail` ==> Imprimer la fin d’un fichier
- `tee` ==> Redirigez le flux standard vers le terminal et vers un fichier
- `wc` ==> Compter le nombre d’éléments de son stdin

find

Rechercher des fichiers dans une hiérarchie de répertoires

Exemple :

```
# find SOMEWHERE -name SOMETHING

user@pem:~$ find /etc/ -name .bashrc
/etc/skel/.bashrc
user@pem:~$ find $HOME -name "*.avi"
/home/user/cours/temp/AF.avi
/home/user/prive/La Bataille de Brest-Litovsk.avi
^C
```

grep

Imprimer des lignes qui correspondent aux motifs

Exemple :

```
user@pem:~$ grep user /etc/passwd
user:x:1000:1000::/home/user:/bin/bash
```

cut

Supprimer des sections de chaque ligne de fichiers

Exemple :

```
user@pem:' -f1  /etc/passwd
root
daemon
bin
[...]
```

xargs

La commande `xargs` permet la construction et l'exécution de lignes de commande à partir de l'entrée standard.

Exemple :

```
user@pem:~$ xargs echo
hello
Word
<CTRL+D>
hello Word
```

or

```
user@pem:~$ xargs mkdir
DirOne
DirTwo
<CTRL+D>
user@pem:~$ ls
DirOne DirTwo
```

Chaînage de commandes

Chaînage simple

Pour lancer plusieurs commandes sur la même ligne :

```
user@pem ~ $ echo test1 ; echo test2
test1
test2
```

Chaînage conditionnel de commandes

Exécuter une commande si la précédente fonctionne

```
user@pem ~ $ true && echo test
test
```

Exécuter une commande si la précédente ne fonctionne pas

```
user@pem ~ $ false || echo test
test
```

Edition sous LINUX/UNIX en mode TEXTUEL

Edition sous LINUX/UNIX en mode TEXTUEL

Sous linux il existe plusieurs traitements de texte en mode textuel. Le mode textuel veut simplement dire édition dans un terminal Ce qui implique pas de menu, pas d'image, pas de mise en forme

Avantages :

```
* Ultra rapide
* Par défaut sur la plupart de distribution UNIX/LINUX
* Portable et multiplateforme
* Accessible depuis le réseau
* Très puissant
* Programmable
* Modulable
* Gestion des fichiers de configuration simplifiée.
```

Inconvenients :

```
* Nécessite de la pratique
* Apprentissage compliqué
* esthétiquement pauvre
```

Editeur de texte type terminal

Voici les editeur texte les plus retrouvés sous Linux et les plus utilisés

- Vi/Vim

Dans ce cours nous nous pencherons sur vim qui est un portage plus accessible de vi

Vim est LE plus utilisé et le plus pratique pour l'édition de code et fichier de configuration.

Editeur de texte type terminal

Voici les editeur texte les plus retrouvés sous Linux et les plus utilisés

- Vi/Vim
- Emacs

Dans ce cours nous nous pencherons sur vim qui est un portage plus accessible de vi

Vim est LE plus utilisé et le plus pratique pour l'édition de code et fichier de configuration.

Editeur de texte type terminal

Voici les editeur texte les plus retrouvés sous Linux et les plus utilisés

- Vi/Vim
- Emacs
- nano

Dans ce cours nous nous pencherons sur vim qui est un portage plus accessible de vi

Vim est LE plus utilisé et le plus pratique pour l'édition de code et fichier de configuration.

Editeur de texte type terminal

Voici les editeur texte les plus retrouvés sous Linux et les plus utilisés

- Vi/Vim
- Emacs
- nano
- jupp / joe

Dans ce cours nous nous pencherons sur vim qui est un portage plus accessible de vi

Vim est LE plus utilisé et le plus pratique pour l'édition de code et fichier de configuration.


Vim

Les Bases de vim

Les modes sous VIM

Sous VIM il existe 3 grands modes qui permettent d'utiliser ce traitement de texte.

```
* le mode interactif
* le mode insertion
* le mode commande
```

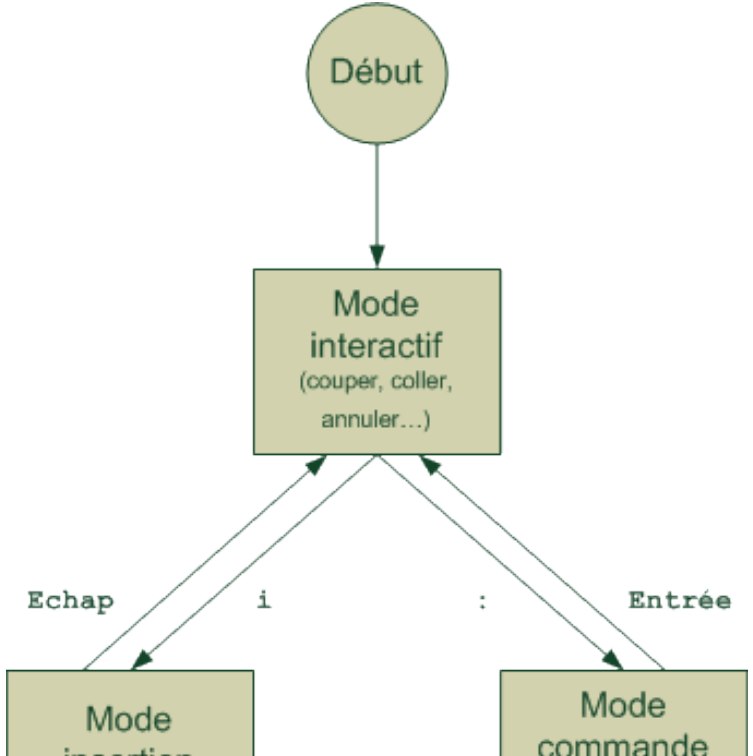
Mode interactif

C'est le mode par défaut par lequel vous commencez en lançant Vim. Vous êtes donc en mode interactif. Dans ce mode, vous ne pouvez pas écrire de texte.

Ce mode vous permet de : copier/coller/supprimer/séléctionner ...

Pour se déplacer sous vi vous utiliserez les touches HJKL qui correspondent respectivement à Gauche Bas Haut Droite

Pour se déplacer sous vim vous utiliserez les flêches Gauche Bas Haut Droite


Début
Mode interactif
(couper, coller, annuler…)
Echap
i
:
Entrée
Mode
Mode commande

Pour les commandes indispensable d'edition du mode interactif :

- v : Permet la sélection d'une partie du texte

Les combos :

Pour les commandes indispensable d'edition du mode interactif :

- v : Permet la sélection d'une partie du texte
- x : Efface le caractère courant (couper)

Les combos :

Pour les commandes indispensable d'edition du mode interactif :

- v : Permet la sélection d'une partie du texte
- x : Efface le caractère courant (couper)
- y : Copie une ligne ou un mot

Les combos :

Pour les commandes indispensable d'edition du mode interactif :

- v : Permet la sélection d'une partie du texte
- x : Efface le caractère courant (couper)
- y : Copie une ligne ou un mot
- d : Couper une ligne ou un mot

Les combos :

Pour les commandes indispensable d'edition du mode interactif :

- v : Permet la sélection d'une partie du texte
- x : Efface le caractère courant (couper)
- y : Copie une ligne ou un mot
- d : Couper une ligne ou un mot
- p : coller

Les combos :

Pour les commandes indispensable d'edition du mode interactif :

- v : Permet la sélection d'une partie du texte
- x : Efface le caractère courant (couper)
- y : Copie une ligne ou un mot
- d : Couper une ligne ou un mot
- p : coller

Les combos :

- yw/dw : copie/coupe le mot

Pour les commandes indispensable d'edition du mode interactif :

- v : Permet la sélection d'une partie du texte
- x : Efface le caractère courant (couper)
- y : Copie une ligne ou un mot
- d : Couper une ligne ou un mot
- p : coller

Les combos :

- yw/dw : copie/coupe le mot
- yy/dd : copie/coupe la ligne courante

Mode insertion

C'est le mode le plus 'classique'. Vous tapez du texte et ce dernier s'insère à l'endroit où se trouve le curseur.

Pour entrer dans ce mode, il existe plusieurs possibilités. L'une des plus courantes est d'appuyer sur la touche i (insertion).

Très important : Pour sortir de n'importe quel mode il faut appuyer sur la touche Echap.

Mode commande

En mode commande vous pouvez aussi

Ce mode est le plus complet et surtout celui qui donne toute la puissance à vim

Il permet de lancer des commandes telles que « quitter », « enregistrer », etc. Vous pouvez aussi l'utiliser pour activer des options de Vim :

```
* coloration syntaxique (:syntax on )

* affichage du numéro des lignes (:set number)

* indenter intelligemment (:set smartindent)
```

Execution d'une commande sous VIM

Tout en étant en mode commande vous pouvez récupérer des informations du shell (la console) telles que ls, locate, cp, etc.

```
* soit en tapant certaines commandes comme :

    ":ls"

    ":locate"

* soit en lançant un sous shell :

    ":sh"
    (pour ce mode une fois la session finie
    avec exit ou CTRL+d le shell rend la main à vim)
```

Pour activer ce mode, vous devez être en mode interactif et appuyer sur la touche deux points « : ». Vous validerez la commande avec la touche Entrée et reviendrez alors au mode interactif.

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

- Enregistrer :

  :w

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

- Enregistrer :

  :w

- Quiter :

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

- Enregistrer :

  :w

- Quiter :

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

- Enregistrer :

  :w

- Quiter :

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

- Enregistrer :

  :w

- Quiter :

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

- Enregistrer :

  :w

- Quiter :

action sur le fichier

En mode commande vous pouvez aussi

- remplacer du texte :

  :s/vieux/nouveau/g

- Trouver du texte :

  /chaine

  pour trouver l'occurrence 'chaine' suivante appuyez sur "n". Quand le document ne trouve plus rien en fin de document il vous indique qu'il va reprendre depuis le début.

- Enregistrer :

  :w

- Quiter :

CHEMIN

$PATH est une liste de répertoires. Avec bash (et non sh), vous n'avez pas à écrire le chemin absolu ou relatif d'une commande. Si la commande que vous tapez existe dans l'un de ces répertoires, bash l'appellera.

```
username@hostname:~$ echo $PATH
/sbin/:/home/username/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/games
:/usr/games
username@hostname:~$ id
uid=1000(username) gid=1000(username) groupes=1000(username)
username@hostname:~$ which id
/usr/bin/id
```

PS1

$PS1 signifie “Prompt String One” ou “Prompt Statement One”

Il s’agit de la première chaine d’invite (que vous voyez sur une ligne de commande). Vous pouvez le modifier facilement “en direct” ou dans votre fichier .bashrc pour qu’il soit effectif dans chaque terminal SHELL

```
username@hostname:~$ echo $PS1
echo $PS1
[e]0;u@h: wa]${debian_chroot:+($debian_chroot)}[033[01;32m]
u@h[033[00m]:[033[01;34m]w[033[00m]$
username@hostname:~$ PS1="Go for it->"
Go for it->echo $PS1
Go for it->
```

TERME

La variable ***$TERM*** définit le type de terminal.

```
username@hostname:~$ echo $TERM
xterm-256color
```

DOMICILE

$HOME est une variable shell bash Linux. Il indique le répertoire personnel de l'utilisateur actuel. Il représente également l'argument par défaut de la commande cd. La valeur de cette variable est également utilisée lors de l'expansion du tilde.

La valeur est définie avec le fichier /etc/passwd au démarrage du système d'exploitation

```
username@hostname:~$ echo $HOME
/home/username
username@hostname:~$ grep username /etc/passwd
username:x:1000:1000:A random user:/home/username:/bin/bash
```

SHELL

Le ***$SHELL*** est une variable d'environnement. Le chemin d'accès complet au shell se trouve dans cette variable d'environnement.

La valeur est définie avec le fichier /etc/passwd au démarrage du système d'exploitation

```
username@hostname:~$ echo $SHELL
/bin/bash
username@hostname:~$ grep username /etc/passwd
username:x:1000:1000:A random user:/home/username:/bin/bash
```

Liste des Linux SHELL

Il y a beaucoup d'autres shell sous linux :

- Shell Bourne (l'antique shell de Steve Bourne) : /bin/sh

Liste des Linux SHELL

Il y a beaucoup d'autres shell sous linux :

- Shell Bourne (l'antique shell de Steve Bourne) : /bin/sh
- Korn shell, le shell de David Korn pour UNIX : /bin/ksh et /bin/pdksh (freeware)

Liste des Linux SHELL

Il y a beaucoup d'autres shell sous linux :

- Shell Bourne (l'antique shell de Steve Bourne) : /bin/sh
- Korn shell, le shell de David Korn pour UNIX : /bin/ksh et /bin/pdksh (freeware)
- Shell C : /bin/csh

Liste des Linux SHELL

Il y a beaucoup d'autres shell sous linux :

- Shell Bourne (l'antique shell de Steve Bourne) : /bin/sh
- Korn shell, le shell de David Korn pour UNIX : /bin/ksh et /bin/pdksh (freeware)
- Shell C : /bin/csh
- Shell Tenex (le shell C s'étend) : /bin/tcsh

Liste des Linux SHELL

Il y a beaucoup d'autres shell sous linux :

- Shell Bourne (l'antique shell de Steve Bourne) : /bin/sh
- Korn shell, le shell de David Korn pour UNIX : /bin/ksh et /bin/pdksh (freeware)
- Shell C : /bin/csh
- Shell Tenex (le shell C s'étend) : /bin/tcsh
- Shell Zorn : /bin/zsh

Liste des Linux SHELL

Il y a beaucoup d'autres shell sous linux :

- Shell Bourne (l'antique shell de Steve Bourne) : /bin/sh
- Korn shell, le shell de David Korn pour UNIX : /bin/ksh et /bin/pdksh (freeware)
- Shell C : /bin/csh
- Shell Tenex (le shell C s'étend) : /bin/tcsh
- Shell Zorn : /bin/zsh
- Bash (Bourne Again Shell, le shell Linux) : /bin/bash

Protéger une variable

IFS

Le ***$IFS*** est un acronyme pour séparateur de champs interne ou séparateur de champs d'entrée. Le $IFS est une variable shell spéciale dans Bash, ksh, sh et POSIX. Voyons ce qu'est $IFS et pourquoi vous devez l'utiliser lors de l'écriture de scripts shell sous Linux et Unix.

Par défaut l'IFS est composé de :

- Retour chariot

IFS peut être modifié avant d'utiliser une commande ou une fonction. Vous pouvez afficher la valeur réelle d'IFS avec la commande suivante

```
username@hostname:$ echo "--$IFS--"
--
--
```

IFS

Le ***$IFS*** est un acronyme pour séparateur de champs interne ou séparateur de champs d'entrée. Le $IFS est une variable shell spéciale dans Bash, ksh, sh et POSIX. Voyons ce qu'est $IFS et pourquoi vous devez l'utiliser lors de l'écriture de scripts shell sous Linux et Unix.

Par défaut l'IFS est composé de :

- Retour chariot
- Tabulation

IFS peut être modifié avant d'utiliser une commande ou une fonction. Vous pouvez afficher la valeur réelle d'IFS avec la commande suivante

```
username@hostname:$ echo "--$IFS--"
--
--
```

IFS

Le ***$IFS*** est un acronyme pour séparateur de champs interne ou séparateur de champs d'entrée. Le $IFS est une variable shell spéciale dans Bash, ksh, sh et POSIX. Voyons ce qu'est $IFS et pourquoi vous devez l'utiliser lors de l'écriture de scripts shell sous Linux et Unix.

Par défaut l'IFS est composé de :

- Retour chariot
- Tabulation
- Espace

IFS peut être modifié avant d'utiliser une commande ou une fonction. Vous pouvez afficher la valeur réelle d'IFS avec la commande suivante

```
username@hostname:$ echo "--$IFS--"
--
--
```

Protéger une variable : Syntaxe

Il y a trois types de “quotes” essentielles dans SHELL :

- guillemet simple ' : la chaine entre ceux-ci ne sera pas interprétée

Protéger une variable : Syntaxe

Il y a trois types de “quotes” essentielles dans SHELL :

- guillemet simple ' : la chaine entre ceux-ci ne sera pas interprétée
- guillemet double " : la chaine entre ceux-ci sera interprétée (caractère spécial comme $)

Protéger une variable : Syntaxe

Il y a trois types de “quotes” essentielles dans SHELL :

- guillemet simple ' : la chaine entre ceux-ci ne sera pas interprétée
- guillemet double " : la chaine entre ceux-ci sera interprétée (caractère spécial comme $)
- backquote ‘ : la chaine entre celles-ci sera une COMMANDE SHELL, vous pouvez utiliser $(COMMAND) pour la rendre plus lisible

Visibilité variable 1/4

Toutes vos variables auront une “portée”, ou une visibilité. Par défaut, les variables définies dans un terminal SHELL sont visibles dans votre terminal SHELL et uniquement dans celui-ci.

```
username@hostname:~$ Var=CONTENU
username@hostname:~$ echo $Var
CONTENU
username@hostname:~$ bash
username@hostname:~$ ps
PID TTY          TIME CMD
12996 pts/2    00:00:00 bash
13009 pts/2    00:00:00 bash
13021 pts/2    00:00:00 ps
username@hostname:~$ echo $Var

username@hostname:~$ exit
```

Visibilité variable 2/4

Vous devrez exporter une variable pour la rendre visible pour un autre terminal ou script SHELL enfant

```
username@hostname:~$ Var=CONTENU
username@hostname:~$ echo $Var
CONTENU
username@hostname:~$ export Var
username@hostname:~$ bash
username@hostname:~$ ps
PID TTY          TIME CMD
12996 pts/2    00:00:00 bash
13406 pts/2    00:00:00 bash
13435 pts/2    00:00:00 ps
username@hostname:~$ echo $Var
CONTENU
```

Visibilité variable 3/4

Il en va de même pour votre script, la portée de votre variable ne sera effective qu'à l'intérieur de votre script Vous pouvez sourcer un autre script pour étendre la visibilité de vos variables au script appelant

Considérez deux scripts Prog1.sh et Prog2.sh comme ci-dessous

```
username@hostname:~$ cat Prog1.sh
#!/bin/bash
Var=CONTENU
echo "Prog1 : $Var"
./Prog2.sh
username@hostname:~$ cat Prog2.sh
#!/bin/bash
echo "Prog2 : $Var"
username@hostname:~$ ./Prog1.sh
Prog1 : CONTENU
Prog2 :
```

À présent ! Considérez deux scripts Prog1.sh et Prog2.sh comme ci-dessous

```
username@hostname:~$ cat Prog1.sh
```

Protéger une variable : Exemples

```
username@hostname:~$ Var="ONE"
username@hostname:~$ echo "$Var"_FILE
ONE_FILE
username@hostname:~$ Var="ONE"
username@hostname:~$ echo '$Var'_FILE
'$Var'_FILE
username@hostname:~$ echo '${Var}'_FILE
${Var}_FILE
username@hostname:~$ echo $Var_FILE
(nohing because Var_FILE dont exist)
username@hostname:~$ echo ${Var}_FILE
ONE_FILE
username@hostname:~$ls
C02 C03 C04 data EXAM Old ORIG
username@hostname:~$ Var=`ls` #or Var=$(ls)
username@hostname:~$ echo $Var
C02 C03 C04 data EXAM Old ORIG
```