

Antony Kervazo-Canut

Bash for Teenagers

**LE MEILLEUR ALLIÉ DU
TERMINAL**



SOMMAIRE

Introduction	3
Navigation & Fichiers	4
Manipulation de Fichiers & Dossiers	5
Redirection	6
Utilisateurs et Groupes	7
Permissions	10
Sudo	11
Visudo	12
Sudoers	13
Alias & Fonctions	14
Prompt personnalisé	15
Awk	16
Scripts Awk	17
Sed	18
Sed - Avancé	19

Introduction



Bash (Bourne Again SHell) est l'interpréteur de commandes le plus couramment utilisé sur les systèmes Linux et UNIX. Il permet aux utilisateurs d'entrer et d'exécuter des commandes et des scripts pour effectuer des opérations sur le système d'exploitation. Bash combine des fonctionnalités utiles de la programmation de scripts avec une interface de ligne de commande puissante, rendant l'administration du système, la gestion de fichiers et l'exécution de tâches automatisées plus efficaces et plus accessibles.

Pourquoi utiliser Bash ?

- **Polyvalent** : Bash est présent sur pratiquement tous les systèmes UNIX et Linux, rendant les scripts écrits en Bash hautement portables.
- **Puissant** : Avec Bash, vous pouvez automatiser presque toutes les tâches liées au système d'exploitation, à la gestion de fichiers, et à la programmation.
- **Accessible** : Contrairement à de nombreux langages de programmation, Bash ne nécessite pas d'environnement de développement spécifique. Vous pouvez commencer avec un simple éditeur de texte et un terminal.

Bash est disponible sur les environnements Linux et MacOS nativement. Sur Windows, il est conseillé d'utiliser WSL pour installer un environnement Linux afin d'y avoir accès.

Navigation & Fichiers



Naviguer dans le système de fichiers, gérer des fichiers et des dossiers, et afficher leur contenu sont des compétences fondamentales pour tout utilisateur de Bash.

```
# Afficher le répertoire actuel
pwd

# Changer de répertoire
cd /chemin/vers/le/dossier

# Remonter d'un niveau dans l'arborescence des dossiers
cd ..

# y compris les fichiers cachés (a), avec des détails tel que
# permissions, propriétaire, taille (l)
ls -la

# Créer un nouveau dossier
mkdir nom_du_dossier

# Créer un nouveau fichier
touch nom_du_fichier

# Copier un fichier ou un dossier
cp source destination

# Déplacer ou renommer un fichier ou un dossier
mv source destination

# Supprimer un fichier
rm nom_du_fichier

# Supprimer un dossier et son contenu
rm -r nom_du_dossier
```

Manipulation de Fichiers & Dossiers



Bash permet des commandes avancées pour la recherche de fichiers et dossiers, l'extraction de contenus spécifiques à partir de fichiers, et la manipulation de l'entrée.

```
● ● ●

# Trouver tous les fichiers et dossiers correspondant à un nom
find /chemin/de/recherche -name nom_du_fichier_ou_dossier

# Trouver des fichiers dans le répertoire actuel et sous-répertoires par nom
find . -name nom_du_fichier

# Trouver des fichiers par type (f pour fichier, d pour dossier)
find /chemin/de/recherche -type f -name nom_du_fichier

# Trouver des fichiers modifiés dans les derniers n jours
find /chemin/de/recherche -type f -mtime -n

# Chercher une chaîne de caractères dans des fichiers
grep 'chaîne_de_caractères' fichier

# Chercher de manière récursive dans un dossier
grep -r 'chaîne_de_caractères' /chemin/du/dossier

# Ignorer la casse lors de la recherche
grep -i 'chaîne_de_caractères' fichier

# Afficher le numéro de ligne avec les résultats de la recherche
grep -n 'chaîne_de_caractères' fichier
```

Redirection



La compréhension et l'utilisation efficace de find, grep, les redirections, et les pipes peuvent considérablement augmenter votre productivité et votre capacité à gérer des tâches liées aux fichiers et au contenu.

```
● ● ●

# Rediriger la sortie d'une commande dans un fichier (écrase le fichier)
commande > fichier

# Ajouter la sortie d'une commande à la fin d'un fichier
commande >> fichier

# Utiliser la sortie d'une commande comme entrée pour une autre
# Utiliser la sortie de 'ls' comme entrée pour 'grep' afin de chercher des fichiers spécifiques
ls | grep 'motif_recherche'

# Rediriger l'erreur standard dans un fichier
commande 2> fichier_erreurs

# Rediriger à la fois la sortie standard et l'erreur standard dans un fichier
commande &> fichier
```

Utilisateurs et Groupes



Chaque fichier et dossier dans un système UNIX ou Linux a un propriétaire et un groupe associé. Ces attributs déterminent qui peut interagir avec le fichier ou le dossier et de quelle manière.

- **Propriétaire (User)** : L'utilisateur qui a créé le fichier ou le dossier. Par défaut, le propriétaire a le contrôle complet sur le fichier, y compris le droit de modifier ses permissions.
- **Groupe (Group)** : En plus d'un propriétaire individuel, chaque fichier est assigné à un groupe. Les utilisateurs appartenant à ce groupe peuvent avoir des permissions distinctes de celles du propriétaire et des autres utilisateurs.

Les permissions déterminent ce que les utilisateurs peuvent faire avec un fichier ou un dossier. Il y a trois types de permissions :

- **Lecture (r, read)** : Permet de voir le contenu d'un fichier ou de lister les fichiers d'un dossier.
- **Écriture (w, write)** : Autorise la modification ou la suppression d'un fichier, ou la modification des contenus d'un dossier (création, suppression de fichiers).
- **Exécution (x, execute)** : Pour les fichiers, cela permet de les exécuter comme des programmes. Pour les dossiers, cela permet de les parcourir et d'accéder aux fichiers qu'ils contiennent.

Utilisateurs et Groupes



Les permissions sont définies séparément pour le propriétaire, le groupe et les autres utilisateurs :

- Propriétaire (u, user) : Les permissions accordées au propriétaire du fichier.
- Groupe (g, group) : Les permissions accordées aux membres du groupe associé au fichier.
- Autres (o, others) : Les permissions accordées à tous les autres utilisateurs du système.

Lorsque vous utilisez la commande `ls -l`, les permissions sont affichées au début de chaque ligne sous forme de 10 caractères. Le premier caractère indique le type de fichier (par exemple, `-` pour un fichier régulier, `d` pour un dossier). Les 9 caractères suivants sont divisés en trois groupes de trois, représentant les permissions pour le propriétaire, le groupe et les autres, dans cet ordre.

Par exemple, `-rwxr-xr--` signifie :

- `-` : Il s'agit d'un fichier régulier.
- `rwx` : Le propriétaire peut lire, écrire et exécuter.
- `r-x` : Les membres du groupe peuvent lire et exécuter, mais pas écrire.
- `r--` : Les autres utilisateurs peuvent uniquement lire.

Pour modifier les permissions, on utilise `chmod`, et pour changer le propriétaire ou le groupe, on utilise `chown`.

Utilisateurs et Groupes



```
# Créer un nouvel utilisateur  
sudo adduser nom_utilisateur  
  
# Modifier les propriétés d'un utilisateur (par exemple, le  
# nom complet)  
sudo usermod -c "Nouveau Nom Complet" nom_utilisateur  
  
# Supprimer un utilisateur (sans supprimer ses fichiers)  
sudo deluser nom_utilisateur  
  
# Supprimer un utilisateur et son répertoire personnel  
sudo deluser --remove-home nom_utilisateur  
  
# Créer un nouveau groupe  
sudo addgroup nom_groupe  
  
# Ajouter un utilisateur existant à un groupe  
sudo adduser nom_utilisateur nom_groupe  
  
# Supprimer un groupe (doit être vide)  
sudo delgroup nom_groupe
```

Permissions



La gestion appropriée des permissions permet de protéger les fichiers contre les accès non autorisés, tandis que la modification des propriétaires et des groupes aide à organiser l'accès aux ressources de manière logique et sécurisée.



```
# Changer les permissions d'un fichier ou dossier  
chmod mode fichier  
  
# Exemples de modes :  
# Ajouter le droit d'exécution au propriétaire  
chmod u+x fichier  
  
# Définir les permissions pour que le propriétaire puisse  
lire et écrire, le groupe lire, et les autres aucun accès  
chmod 640 fichier  
  
# Utiliser le mode symbolique pour donner à tous les  
utilisateurs le droit de lire un fichier  
chmod a+r fichier  
  
# Changer le propriétaire d'un fichier  
chown nouveau_PROPRIETAIRE fichier  
  
# Changer le groupe d'un fichier  
chown :nouveau_GROUPE fichier  
  
# Changer à la fois le propriétaire et le groupe  
chown nouveau_PROPRIETAIRE:nouveau_GROUPE fichier
```

Sudo



sudo permet à un utilisateur autorisé d'exécuter des commandes avec les priviléges du superutilisateur ou d'un autre utilisateur, tel que spécifié dans le fichier de configuration sudoers.

Le fichier /etc/sudoers contrôle les permissions sudo. Il est recommandé d'utiliser visudo pour éditer ce fichier afin d'éviter les erreurs de syntaxe qui pourraient rendre le système inutilisable.

```
# Exécuter une commande en tant que superutilisateur  
sudo commande  
  
# Éditer le fichier sudoers de manière sécurisée  
sudo visudo
```

Visudo



Comprendre l'utilisation de visudo et la configuration du fichier sudoers est crucial pour gérer efficacement les priviléges d'administration dans les systèmes UNIX et Linux. visudo est l'éditeur de texte sécurisé pour le fichier sudoers, et il vérifie les erreurs de syntaxe avant d'appliquer les changements, ce qui réduit le risque de configuration incorrecte pouvant compromettre la sécurité ou la fonctionnalité du système.

Lorsque vous lancez visudo, il ouvre le fichier sudoers dans l'éditeur de texte par défaut du système, souvent vi ou nano, selon la distribution.

Commandes de Base pour Vi/Vim

Si vi ou vim est l'éditeur utilisé par visudo :

- Insérer du Texte : Appuyez sur i pour passer en mode insertion, puis commencez à taper.
- Sauvegarder et Quitter : Appuyez sur Esc pour quitter le mode insertion, tapez :wq, puis appuyez sur Enter.
- Quitter sans Sauvegarder : Appuyez sur Esc, tapez :q!, puis appuyez sur Enter.

Commandes de Base pour Nano

Si nano est l'éditeur utilisé par visudo :

- Insérer du Texte : Déplacez le curseur avec les touches fléchées et commencez à taper.
- Sauvegarder : Appuyez sur Ctrl + O, puis Enter.
- Quitter : Appuyez sur Ctrl + X. Si vous n'avez pas sauvégarde, il vous demandera si vous voulez le faire.

Sudoers



```
# Accorder les Droits Sudo à un Utilisateur
nom_utilisateur ALL=(ALL:ALL) ALL

# Accorder les Droits Sudo à un Groupe / le symbole %
# indiquant un groupe.
%nom_groupe ALL=(ALL:ALL) ALL

# Exécution de Commandes Sans Mot de Passe
nom_utilisateur ALL=(ALL:ALL) NOPASSWD: ALL

# Restriction des Commandes
nom_utilisateur ALL=(ALL:ALL) /chemin/vers/commande1,
/chemin/vers/commande2
# nom_utilisateur est seulement autorisé à exécuter commande1
et commande2 avec sudo

# Configurer un Alias de Commande
Cmnd_Alias WEB = /etc/init.d/apache, /etc/init.d/nginx
nom_utilisateur ALL=(ALL:ALL) NOPASSWD: WEB
# Définit un alias WEB pour un ensemble de commandes et
permet à nom_utilisateur de les exécuter sans mot de passe.
```

Toujours utiliser visudo pour éditer le fichier sudoers pour éviter des erreurs de syntaxe qui pourraient rendre le système inutilisable.
Éviter d'accorder des privilèges sudo illimités à moins que cela soit absolument nécessaire pour réduire les risques de sécurité.
Tester les nouvelles règles avec précaution pour s'assurer qu'elles fonctionnent comme prévu sans compromettre la sécurité.

Alias & Fonctions



La personnalisation de votre environnement Bash peut améliorer considérablement votre productivité et votre confort lors de l'utilisation de la ligne de commande.

Les alias et les fonctions permettent de créer des raccourcis pour des commandes longues ou complexes que vous utilisez fréquemment.

Pour rendre les alias permanent, ajoutez-le à votre fichier `~/.bashrc` ou `~/.bash_profile`.

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

meteo() {
    curl http://v2.wttr.in/lansing
}
```

`.bash_profile` (ou `.profile` sur certaines distributions) est exécuté pour les shells de login. Il est courant d'y inclure le `.bashrc` pour appliquer les mêmes paramètres aux deux types de shells.

```
if [ -f ~/.bashrc ]; then
    source ~/.bashrc
fi
```

Prompt personnalisé



La personnalisation du prompt se fait en modifiant la variable d'environnement PS1 et PROMPT si le prompt est zsh (pour les versions récentes de macOS). Cette configuration fournira un prompt coloré affichant le nom de l'utilisateur, le nom de l'hôte, le répertoire courant, ainsi que l'heure actuelle de manière cohérente à travers les deux shells.

Pour les systèmes linux, il faudra modifier le fichier ~/.bashrc. Pour macos selon le type de terminal, il faudra modifier le fichier ~/.bash_profile ou ~/.zshrc. Bien sûr, si le fichier n'existe pas, il faudra le créer.

```
● ● ●

# Personnalisation du prompt bash
export PS1="\[\e[32m\]\u@\h\[\e[36m\] \w \
[\e[35m\]\D{\%H:\%M:\%S}\[\e[00m\] \$ "

# Personnalisation du prompt zsh
PROMPT='%F{green}%n@%m %F{cyan}%~ %F{magenta}%D{\%L:\%M:\%S}%
%f'
```

Il faudra ensuite sourcer le fichier (commande source nom_fichier) pour appliquer ses changements. De nombreuses personnalisations en ligne existent, il suffira de fouiller un peu.

Awk



awk est un outil de manipulation de données extrêmement puissant et flexible, particulièrement adapté à la manipulation de textes et de données structurées.

awk fonctionne en lisant un fichier ou un flux ligne par ligne, en divisant chaque ligne en champs, et en appliquant des actions spécifiées par l'utilisateur sur ces champs. Les champs sont par défaut séparés par des espaces blancs, mais ce séparateur peut être personnalisé.

```
# Afficher les lignes où le 3e champ est supérieur à 100
awk '$3 > 100' fichier.txt

# Somme des valeurs d'un champ
awk '{sum += $2} END {print sum}' fichier.txt

# Concaténation de champs
awk '{print $1, $2, $3 " " $4}' fichier.txt

# Nombre de lignes, de champs et nom du fichier en cours
awk '{print NR, NF, FILENAME}' fichier.txt

# Afficher la première et la dernière ligne
awk 'NR==1; END {print}' fichier.txt

# Spécifier un séparateur de champs personnalisé (une virgule)
awk -F , '{print $1}' fichier.csv
```

Scripts Awk



Pour des tâches plus complexes, vous pouvez écrire des scripts .awk peut inclure des fonctions, des boucles, etc, le rendant aussi puissant qu'un langage de programmation.

```
● ● ●

# premier fichier .awk
#!/usr/bin/awk -f
BEGIN {print "Début du traitement"}
{sum += $2}
END {print "Somme : ", sum}

# second fichier .awk
#!/usr/bin/awk -f
BEGIN {
    print "Début du traitement des données"
}

{
    somme += $1
    compteur++
}

END {
    print "Moyenne : ", somme / compteur
}
```

```
# Exécute un fichier .awk sur un fichier texte
awk -f fichier.awk fichier.txt
```

Sed



sed (Stream Editor) est un outil puissant pour manipuler le texte dans les flux de données et les fichiers. Utilisé principalement pour l'extraction de texte, la substitution, et la suppression, sed fonctionne en lisant l'entrée ligne par ligne, en appliquant des opérations spécifiées, et en affichant ou en modifiant le texte.

```
● ● ●

# Pour chaque ligne, remplace la première occurrence d'un
# fichier
sed 's/ancien/nouveau/' fichier

# Remplacer toutes les occurrences d'un fichier
sed 's/ancien/nouveau/g' fichier
# L'option g indique à sed de remplacer toutes les
# occurrences trouvées dans une ligne, de chaque ligne du
# fichier

# Substitution avec Adresse
sed '2,4s/ancien/nouveau/' fichier
# Remplace ancien par nouveau seulement entre les lignes 2 et
# 4.

# Supprimer une ligne spécifique
sed '3d' fichier
# Supprime la troisième ligne du fichier.

# Supprimer des lignes correspondant à un motif
sed '/motif/d' fichier
# Supprime toutes les lignes contenant motif.

# Ajouter une Ligne Avant une Ligne Spécifique
sed '3i\nouvelle ligne' fichier
# Insère "nouvelle ligne" avant la ligne 3. utiliser "a"
# permet d'ajouter une ligne après
```

Sed - Avancé



sed supporte des expressions régulières étendues, offrant une grande flexibilité pour la manipulation de texte.

```
# Plusieurs motifs remplacés
sed -r 's/(motif1|motif2)/remplacement/g' fichier

# Modifie le fichier et crée une copie de sauvegarde de
l'original avec l'extension .bak.
sed -i'.bak' 's/ancien/nouveau/g' fichier

# Réarrange la date en format jour-mois-année.
echo "La date est 2023-04-15" | sed 's/\(20[0-9][0-9]\)-\([0-
1][0-9]\)-\([0-3][0-9]\)\)/\3-\2-\1/'

# Chaine plusieurs commande sed a la suite
sed -e 's/ancien/nouveau/g' -e '/motif/d' fichier.txt
# Remplace "ancien" par "nouveau" sur toutes les lignes puis
supprime toutes les lignes contenant "motif".

# Chaine plusieurs commande sur un format multiligne
sed -e '
s/ancien/nouveau/g
/motif/d
' fichier.txt

# Place de modification
sed '/début/,/fin/s/ancien/nouveau/g' fichier.txt
# Remplace "ancien" par "nouveau" entre "début" et "fin".

# Modification conditionnelle
sed '/motif/{s/ancien/nouveau/g;}' fichier.txt
Si une ligne contient "motif", alors "ancien" est remplacé
par "nouveau" sur cette ligne.

# Ajoute une ligne avant si celle-ci contient un motif,
utiliser a pour "après"
sed '/motif/i Nouvelle ligne avant' fichier.txt
```

Dans la même collection

Orchestration et Gestion de Conteneurs



Infrastructure as Code



Sécurité & Gestion des secrets



Développement & CI/CD



↓ FOLLOW ME ↓



[ANTONYCANUT](#)



[ANTONY KERVAZO-CANUT](#)