Tutoriel BASH pour Débutants : Introduction aux Concepts Fondamentaux

Introduction (2 minutes)

Ce tutoriel présente les bases de **BASH** (Bourne Again SHell), l'interface de commande par défaut dans la plupart des systèmes Linux. BASH permet d'interagir avec le système d'exploitation en saisissant des instructions dans un terminal. Ce cours explique comment naviguer dans les fichiers, manipuler des données, utiliser des variables d'environnement, comprendre les caractères spéciaux et utiliser la commande `grep`. Chaque notion est accompagnée d'exemples pratiques pour illustrer son utilisation.

Le terminal, accessible via `Ctrl + Alt + T` ou en recherchant "Terminal" dans le menu, affiche une invite de commande (par exemple, `utilisateur@machine:~$`). C'est l'interface où les commandes BASH sont exécutées.

1. Navigation dans le Système de Fichiers (2 minutes)

BASH permet de naviguer dans l'arborescence des fichiers et dossiers. Voici les commandes essentielles :

- **`pwd`** (Print Working Directory) : Affiche le chemin absolu du dossier courant.
  Utilisation : Exécutez `pwd` pour connaître votre position, par exemple `/home/utilisateur`.

- **`ls`** (List) : Liste les fichiers et dossiers dans le répertoire courant.
  Utilisation : Exécutez `ls` pour voir le contenu. Utilisez `ls -l` pour des informations détaillées (permissions, taille, date).

- **`cd`** (Change Directory) : Modifie le répertoire courant.
  Utilisation :
  - `cd Documents` pour entrer dans le dossier "Documents".
  - `cd ..` pour remonter au dossier parent.
  - `cd` seul pour revenir au répertoire personnel (`/home/utilisateur`).

Exemple : Exécutez `ls`, puis `cd Documents`, et enfin `ls -l` pour explorer le contenu détaillé du dossier "Documents".

2. Variables d'Environnement (4 minutes)

Les **variables d'environnement** sont des paramètres qui stockent des informations utilisées par BASH et les programmes. Elles configurent le comportement du système et sont accessibles dans le terminal.

- **Principales variables d'environnement** :
  - `HOME` : Contient le chemin du répertoire personnel de l'utilisateur, par exemple `/home/utilisateur`. Utilisé pour localiser les fichiers personnels.
  - `PATH` : Liste les répertoires où BASH recherche les commandes exécutables, comme `/usr/bin:/bin`. Permet d'exécuter des programmes sans spécifier leur chemin complet.
  - `USER` : Stocke le nom de l'utilisateur actuel, par exemple `utilisateur`. Utile pour personnaliser des scripts ou affichages.
  - `SHELL` : Indique le shell utilisé, généralement `/bin/bash`. Définit l'environnement de commande actif.
    Exemple : Exécutez `echo $HOME` pour voir `/home/utilisateur`, ou `echo $PATH` pour voir les répertoires de commandes.
- **Afficher une variable** : Utilisez `echo` suivi du nom de la variable précédé de `$`.
  Exemple : `echo $USER` affiche le nom de l'utilisateur actuel.
- **Créer une variable** : Assignez une valeur avec `nom=valeur`.
  Exemple :
  - `MON_NOM=Alice` crée une variable `MON_NOM`.
  - Exécutez `echo $MON_NOM` pour afficher "Alice".
    Note : Les variables créées ainsi sont temporaires et disparaissent à la fermeture du terminal.
- **Exporter une variable** : Utilisez `export` pour rendre une variable accessible aux programmes lancés dans le terminal.
  Exemple :
  - `export MA_VARIABLE=test` définit une variable accessible aux scripts ou processus.
  - Vérifiez avec `printenv MA_VARIABLE`.

Exemple : Exécutez `echo $HOME` pour voir votre répertoire personnel, puis créez une variable avec `MON_TEST=bonjour` et affichez-la avec `echo $MON_TEST`.

3. Manipulation des Fichiers et Répertoires (2 minutes)

BASH permet de créer, modifier et supprimer des fichiers et dossiers.

- **`mkdir`** (Make Directory) : Crée un répertoire.
  Utilisation : `mkdir Projet` crée un dossier nommé “Projet”. Vérifiez avec `ls`.

- **`touch`** : Crée un fichier vide.
  Utilisation : `touch note.txt` crée un fichier texte vide. Vérifiez avec `ls`.

- **`cat`** : Affiche le contenu d’un fichier.
  Utilisation : Exécutez `echo "Exemple" > fichier.txt`, puis `cat fichier.txt` pour voir “Exemple”.

- **`rm`** (Remove) : Supprime des fichiers ou dossiers.
  Utilisation :
  - `rm fichier.txt` supprime un fichier.
  - `rm -r Projet` supprime un dossier et son contenu.
    Attention : Les suppressions sont irréversibles.

Exemple : Créez un dossier avec `mkdir Test`, un fichier avec `touch test.txt`, écrivez avec `echo "Salut" > test.txt`, et lisez avec `cat test.txt`.

4. Caractères Spéciaux dans BASH (3 minutes)

Les **caractères spéciaux** contrôlent le comportement des commandes. Voici les principaux :

- **Redirection (`>` et `>>`)** : Redirige la sortie d’une commande vers un fichier.
  - `>` remplace le contenu d’un fichier.
    Exemple : `echo "Nouveau texte" > fichier.txt` crée ou remplace le contenu par “Nouveau texte”.
  - `>>` ajoute à la fin du fichier.
    Exemple : `echo "Ligne supplémentaire" >> fichier.txt` ajoute sans écraser.

- **Pipe (`|`)** : Envoie la sortie d’une commande comme entrée à une autre.
  Exemple : `ls | grep txt` affiche uniquement les fichiers contenant “txt” dans leur nom.

- **Caractère de substitution (`*`)** : Représente n’importe quelle chaîne de caractères.
  Exemple : `ls *.txt` liste tous les fichiers se terminant par “.txt”.

- **Guillemets (`"` et `'`)** : Protègent les caractères spéciaux ou espaces.

- Les guillemets doubles (`"`) permettent l'expansion des variables.
  Exemple : `echo "Mon chemin : $HOME"` affiche le chemin du répertoire personnel.
- Les guillemets simples (`'`) désactivent l'expansion.
  Exemple : `echo 'Mon chemin : $HOME'` affiche littéralement "Mon chemin : $HOME".

Exemple : Exécutez `echo "Utilisateur : $USER" > info.txt` , puis `cat info.txt` pour voir le résultat.

5. Utilisation de la Commande `grep` (2 minutes)

La commande **`grep`** (Global Regular Expression Print) permet de rechercher du texte dans des fichiers ou dans la sortie d'autres commandes. Elle est particulièrement utile pour filtrer des informations spécifiques.

- **Recherche dans un fichier** : `grep "mot" fichier.txt` affiche les lignes contenant "mot" dans `fichier.txt` .
  Exemple : Créez un fichier avec `echo -e "Ligne 1\nLigne 2\nTest" > exemple.txt` , puis exécutez `grep "Ligne" exemple.txt` pour afficher les lignes contenant "Ligne".
- **Recherche avec une pipe** : Combine `grep` avec une autre commande via `|` .
  Exemple : `ls | grep txt` affiche uniquement les fichiers dont le nom contient "txt".
- **Options utiles** :
  - `-i` : Ignore la casse (majuscules/minuscules). Exemple : `grep -i "test" fichier.txt` .
  - `-n` : Affiche les numéros de ligne. Exemple : `grep -n "Ligne" exemple.txt` .

Exemple : Créez un fichier `test.txt` avec plusieurs lignes, puis utilisez `grep "mot" test.txt` pour trouver un mot spécifique.

6. Conclusion et Ressources (1 minute)

Ce tutoriel a couvert la navigation dans les fichiers, les variables d'environnement, la manipulation de fichiers, les caractères spéciaux et l'utilisation de `grep` dans BASH. Ces compétences permettent de contrôler efficacement un système Linux via le terminal.

Pour approfondir :

- Consultez l'aide des commandes avec `man` (par exemple, `man grep`).

- Explorez des commandes comme `find` (recherche de fichiers) ou `chmod` (gestion des permissions).
- Pratiquez régulièrement pour maîtriser BASH.

Exercice final : Créez un dossier `Exercice`, entrez-y avec `cd Exercice`, créez un fichier avec `touch info.txt`, écrivez le contenu de `$HOME` dedans avec `echo $HOME > info.txt`, et cherchez “home” dans le fichier avec `grep "home" info.txt`.