# BASH - Découvrir son système d'exploitation

[Mise à jour le 2/9/2022]

**En cours de rédaction**

- **Sources**
  - Wiki Ubuntu-fr
    - Gestion des utilisateurs et des groupes en ligne de commande
    - Gérer les droits d'accès (propriétés et permissions) des fichiers et répertoires
- **Ressources**
  - Manuel Linux en Français
  - Initiez-vous à Linux
  - Structure des fichiers et des dossiers
  - La console
  - Commandes Linux

## Introduction

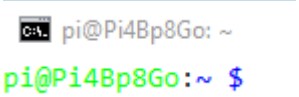
Le **shell** désigne un **interpréteur de lignes de commande**. Lorsque l'invite de commande s'affiche, l'utilisateur peut saisir une nouvelle ligne de commande. La ligne de commande regroupe **une ou plusieurs commandes** et se termine par un **retour à la ligne**.
Une **commande** est composée d'un **nom** qui décrit une action ou un programme, parfois suivie d'**arguments** qui précisent les **paramètres** de l'action à effectuer.

*.bash

```
# Exemple
$ cal -m apr # nom : cal, option : m, argument d'option: apr
```

> Le shell Bash est en permanence associé à un répertoire dans lequel s'exécutent les commandes. Ce répertoire est appelé **répertoire courant**.

## 1. La ligne de commande



### 1.1 L'invite de commande

Lorsque l'on démarre un terminal sur lequel le shell Bash est présent, on est accueilli avec l'**invite de commande** appelée **prompt** en anglais.

***Organisation***

*nom_utilisateur@nom_hôte:répertoire_de_travail prompt*

***Exemple***

*eleve1@Pi4mno:~ $*

| | | |__ prompt : **$** ⇔ normal (non-administrateur), **#** ⇔super utilisateur (root)
| | | ____ répertoire de travail : **~** ⇔ répertoire personnel
| | _____ nom de l'hôte
| _____nom de l'utilisateur


### 1.2 Les commandes (généralités)

Le shell est une application qui sert d'**interface entre le noyau du système d'exploitation et l'utilisateur**. Il sert à exécuter des commandes qui proviennent d'un terminal (**mode interactif**) ou d'un fichier (**mode script**). Ces commandes peuvent être **internes** ou **externes** au shell. Les commandes externes font appel à des programmes séparés du shell tandis que les commandes internes sont exécutées par le shell.

*.bash

```
# La commande interne type suivie du nom d'une commande sert à indiquer le type de la commande
pi@Pi4Bp8Go:~ $ type echo man
echo est une primitive du shell # type built-in (commande interne)
man est /usr/bin/man # commande externe dans le répertoire /usr/bin
```

Une commande (interne ou externe) est constituée par des mots séparés par des espaces. Le nombre d'arguments dépend de la commande et de l'action à effectuer par la commande.

> *Format* : **commande arg1 arg2** … **argn**

*Exemples* : la commande date avec et sans arguments

*.bash

```
pi@Pi4Bp8Go:~ $ date
mercredi 28 avril 2021, 17:16:16 (UTC+0200)

pi@Pi4Bp8Go:~ $ date +%s
1619623002 # nombre de secondes écoulées depuis le 1e premier janvier 1970
```

Un argument peut être une **option**, il sera alors de la forme **-x** avec x la lettre identifiant l'option. Une lettre étant peu explicite, il est souvent possible d'identifier une option via **un ou plusieurs mots**. Elle sera alors préfixée de deux tirets hauts **--**.

*Exemples* : la commande date avec une option

*.bash

```
pi@Pi4Bp8Go:~ $ date -u
mercredi 28 avril 2021, 15:24:03 (UTC+0000)

pi@Pi4Bp8Go:~ $ date --utc
mercredi 28 avril 2021, 15:24:10 (UTC+0000)
```

Pour qu'elle ait un sens, une **option** doit parfois être suivie d'une **valeur**, appelée **argument d'option**.

*Exemples* : la commande cal (CALendar)

*.bash

```
pi@Pi4Bp8Go:~ $ cal -m apr # option: -m, valeur : apr
     Avril 2021
di lu ma me je ve sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

La valeur associée à l'option peut être spécifiée dans le même argument, mais séparée de l'identifiant d'option via un **caractère délimiteur**.

*Exemple* : la commande date

*.bash

```
pi@Pi4Bp8Go:~ $ date --date="2021-04-28" # caractère délimiteur: =
```

```
mercredi 28 avril 2021, 00:00:00 (UTC+0200)
```

La norme **POSIX** (**P**ortable **O**perating **S**ystem **I**nterface) répond au besoin de compatibilité entre les systèmes d'exploitation qui se trouvent sur les différents équipements informatiques. Elle définit diverses interfaces d'outils, de commandes et d'interfaces pour la programmation en **langage C**.

## 2. Trouver de l'aide

La documentation est fournie par les commandes elles-mêmes et l'utilisateur peut y accéder soit au travers de la commande, soit au travers d'utilitaires dédiés.

- **Aide depuis la commande**

La plupart des commandes disposent d'une **aide-interne** accessible avec **--help** ou **-h**.

*Exemple*

*.bash

```
pi@Pi400mno:~ $ date --help
Utilisation : date [OPTION]... [+FORMAT]
         ou : date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

- **La commande man**

Les systèmes de type Unix disposent d'un outil de visualisation des manuels appelé **man** (MANual). La commande man interprète des fichiers de documentation 1 puis les affiche via le lecteur de fichier **less**.

*Exemple*

*.bash

```
pi@Pi400mno:~ $ man date

DATE(1)                                                User Commands
DATE(1)

NAME
       date - print or set the system date and time
```

```
SYNOPSIS
       date [OPTION]... [+FORMAT]
       date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]

DESCRIPTION
       Display the current time in the given FORMAT, or set the system
date. etc.
```

- **La commande apropos**

> **apropos** permet de lister les manuels dont la description comprend les mots passés en argument et donc de trouver une commande dont on ne connaît pas le nom.

*Exemple*

*.bash

```
pi@Pi400mno:~ $ apropos encoding # recherche d'un convertisseur
d'encodage

bind_textdomain_codeset (3) - set encoding of message translations
chardet (1)          - universal character encoding detector
chardet3 (1)         - universal character encoding detector
...
iconv (1)            - convert text from one character encoding to
another # c'est ce que l'on cherche
```

- **La commande whatis**

> L'affichage de la description courte d'une commande s'obtient par la commande **whatis**.

*Exemple*

*.bash

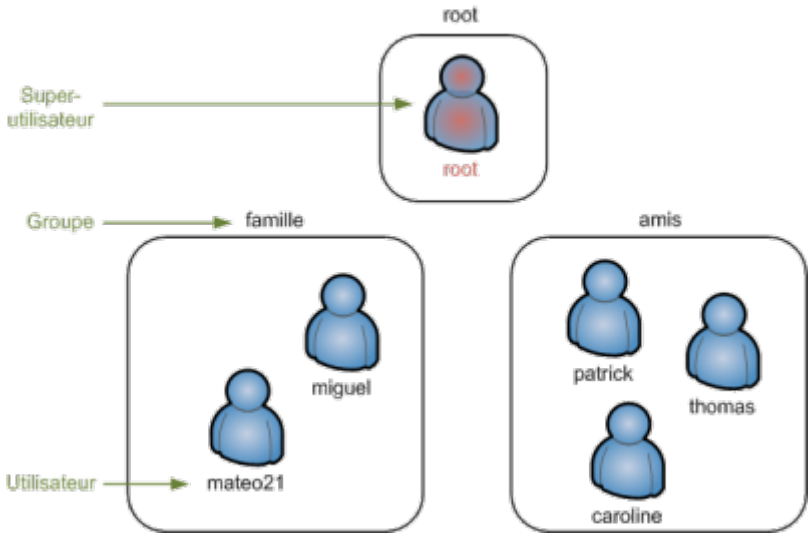
```
pi@Pi400mno:~ $ whatis apropos

apropos (1)          - Chercher le nom et la description des pages de
manuel
```

## 3. Système de fichiers et répertoire

## 4. Les utilisateurs, les groupes et les droits

> Linux est un système **multi-utilisateur**. Cela signifie que plusieurs personnes peuvent travailler simultanément sur le même OS, en s'y connectant à distance notamment.

- **Organisation des utilisateurs sous Linux**




- **Devenir super utilisateur**

> Pour passer de simple utilisateur à super utilisateur, on utilise la commande **sudo** (**S**ubstitute **U**ser **DO**) puis on entre le mot de passe super utilisateur.

*Exemple*

*.bash

```
sudo commande
sudo su commande # su pour rester super utilisateur
```

> **ATTENTION**
> Seul le super utilisateur (**root**) peut créer des utilisateurs et des groupes.

### 4.1 Les utilisateurs

- **Source**
  - Wiki Ubuntu-fr : Gestion des utilisateurs et groupes en ligne de commande

#### Créer un utilisateur

> Lorsqu'on crée un utilisateur, le répertoire personnel portant son nom est automatiquement créé : **/home/***nom*. Un mot de passe est demandé. Pour des raisons de sécurité, ce qui est entré au clavier n'apparaît pas à l'écran.

*Commande*

*.bash

```bash
sudo adduser nom_utilisateur
```

#### Supprimer un utilisateur

*Commandes*

*.bash

```bash
sudo deluser nom_utilisateur # Supprime l'utilisateur, mais pas son répertoire personnel
sudo deluser --remove-home nom_utilisateur # Supprime l'utilisateur et son répertoire personnel
```

#### Changer le mot de passe d'un utilisateur

*Commande*

*.bash

```bash
sudo passwd nom_utilisateur
```

#### Afficher la liste des utilisateurs et des groupes

- **Source** : tuto.eu

> Le fichier **/etc/passwd** contient toutes les informations relatives aux utilisateurs (login, mots de passe, ...).

*Commande*

*.bash

```
grep bash /etc/passwd | cut -f1 -d:
```

### 4.2 Les groupes

> Chaque utilisateur appartient à un groupe. Si on ne définit rien, un groupe du même nom que l'utilisateur est automatiquement créé.

*Exemple*

*.bash

```

```

#### Créer un groupe

*Exemple*

*.bash

```
sudo addgroup nom_groupe
```

*Exemple*

*.bash

```
sudo addgroup www-data
```

#### Mettre un utilisateur dans un ou plusieurs groupes

La commande **usermod** permet d'éditer un utilisateur. On utilise les paramètres :

- **-l** pour renommer l'utilisateur (il faudra également renommer son répertoire)
- **-g** pour changer le groupe
- **-G** pour mettre un utilisateur dans plusieurs groupes

*Commande*

*.bash

```
sudo usermod -g nom_groupe nom_utilisateur # place l'utilisateur nom_utilisateur dans le groupe nom_groupe
sudo usermod -G goupe1,groupe2,groupe3 nom_utilisateur # place l'utilisateur nom_utilisateur dans
                                                       # les groupes group1, groupe2 et groupe3
```

*Exemple*

*.bash

```
sudo usermod -g www-data ruche1
```

#### Supprimer un groupe

*Commande*

*.bash

```
delgroup nom_groupe
```

#### Afficher la liste des groupes

*Exemple*

*.bash

```
cat /etc/group | awk -F: '{print $ 1}'
```

### 4.3 Gestion des droits

- **Source**
  - Wiki Ubuntu-fr - Gérer les droits d'accès (propriétés et permissions) des fichiers et des

répertoires

## 5.1 Traitement des fichiers texte