Les bases d'Unix

Julien Baste

IUT de Lille

Séance 01

2025/2026

Le shell

Sous Unix, on communique avec le système grâce à un interpreteur de commandes (shell).
Un shell est un language de commandes.

Ce programme utilisateur :

1. Affiche un message d'invite (*prompt*)
   En général : `<login>@<machine>:~$`
2. Attend la validation d'une ligne de commandes (touche Entrée)
3. Interprète (traduit et exécute) les commandes données
4. Retourne en 1

Le shell peut être utilisé au travers du terminal. C'est le mode dit "interactif".

Les shells

Il existe plusieurs shells :

- Le Thompson shell (1971) et le Bourne shell (1977) -> dit shell
  - American Telephone & Telegraph
- Le Bourne-Again shell (1988) -> dit bash.
  - Fait partie du projet GNU.
  - Le plus utilisé
  - Celui que nous utiliserons

Mais aussi :

- Le Korn SHell
- Le Z Shell
- Le (Debian) Almquist shell
- Le C-shell
- Le Tenex C-SHell

Syntaxe générale des commandes UNIX

Les commandes sont de la forme :

```
commande [options...] [arguments...]
```

Une commande peut être suivie de *paramètres* :

- des *options* → COMMENT
  - pour préciser son fonctionnement
  - mot commençant généralement par le caractère -

Syntaxe générale des commandes UNIX

Les commandes sont de la forme :

```
commande [options...] [arguments...]
```

Une commande peut être suivie de *paramètres* :

- des *options* → COMMENT
  - pour préciser son fonctionnement
  - mot commençant généralement par le caractère -
- des *arguments* → QUOI
  - pour spécifier des éléments que la commande doit prendre en compte
  - généralement pour identifier des fichiers

Syntaxe générale des commandes UNIX

Les commandes sont de la forme :

```
commande [options...] [arguments...]
```

Une commande peut être suivie de *paramètres* :

- des *options* → COMMENT
  - pour préciser son fonctionnement
  - mot commençant généralement par le caractère -
- des *arguments* → QUOI
  - pour spécifier des éléments que la commande doit prendre en compte
  - généralement pour identifier des fichiers

Beaucoup de libertés :

- chaque commande décide de sa syntaxe ..... *nature et ordre des paramètres*
- une ligne peut comporter plusieurs commandes ..... *séparation par ;*

Documentation

`man` : manuel du système.

- Une page par programme ou fonction.

Chaque page est découpée en plusieurs sections

NAME	le nom et une description rapide de la commande
SYNOPSIS	toutes les possibilités de saisies liées à cette commande (syntaxe)
DESCRIPTION	une explication des conséquences de la commande
FILES	les fichiers modifiés par la commande ou nécessaires au moment de la saisie
OPTIONS	la liste des différentes options de cette commande
SEE ALSO	les références croisées vers d'autres commandes proches
DIAGNOSTICS	des explications sur les messages d'erreur
RETURN VALUES	ce que renvoie la commande
BUGS	des problèmes connus de cette commande
EXAMPLES	des exemples d'appel à cette commande
TIPS	des astuces pour utiliser cette commande.

Documentation

`man` : manuel du système.

- Une page par programme ou fonction.

Chaque page est rangée dans une **section**

1. Executable programs or shell commands
2. System calls (functions provided by the kernel)
3. Library calls (functions within program libraries)
4. Special files (usually found in /dev)
5. File formats and conventions eg /etc/passwd
6. Games
7. Miscellaneous (including macro packages and conventions)
8. System administration commands (usually only for root)
9. Kernel routines [Non standard]

Pour faire référence à une section particulière on la spécifie entre parenthèses `man(1), info(1)`

Système de fichiers

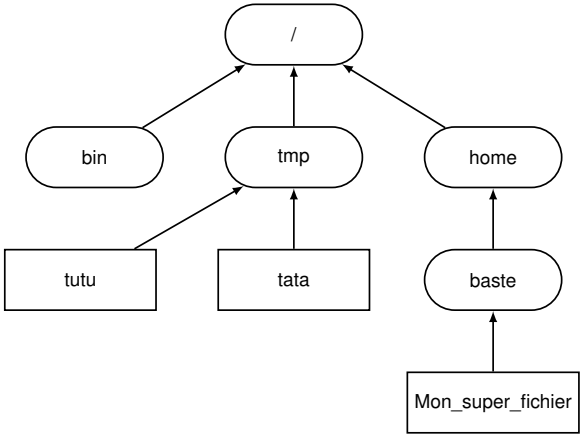
Sous Unix TOUT EST FICHIER . . . ou presque

- En interne (vue du noyau)
  - les fichiers ont tous la même structure
- En externe (vue de l'utilisateur)
  - ordinaires (ou réguliers)
  - dossiers (ou répertoires ou catalogues)
  - liens symboliques
  - tubes
  - sockets
  - spéciaux

Des dossiers pour ranger ces fichiers

- Un dossier est un fichier qui “contient” d’autres fichiers (ordinaires, dossiers, ou autre).
- Chaque fichier est stocké dans un dossier (le père du fichier).
  - Le dossier nommé "`/`" (qui se prononce racine) est la seule exception.
- Tous les fichiers ont "`/`" comme ancêtre.
  - Soit son père
  - Soit le père de son père
  - Soit le père du père de son père
  - etc.
- On peut identifier un fichier par l’ensemble des dossiers qu’il faut traverser pour l’atteindre a partir de la racine (de "`/`"). Cet identifiant s’appelle le chemin absolu.

Arborescence de fichiers



Le chemin absolu est le chemin allant de la racine (“/”) jusqu’au fichier ciblé.

Chemin absolu de `Mon_super_fichier` : `/home/baste/Mon_super_fichier`

Chemin absolu de `tata` : `/tmp/tata`

Où suis-je ?

Navigateur de fichier :

- Affiche le contenu d'un dossier
- Indique quel dossier il vous présente. (Dossier de travail courant)

Avec le terminal c'est pareil

- `pwd` → montre dans quel dossier vous êtes. (Dossier de travail courant)
- `ls` → montre les fichiers contenus dans le dossier.

On peut aussi utiliser :

- `cd` → permet de changer le dossier de travail courant
- `mkdir` → permet de créer un dossier
- `rmdir` → permet de supprimer un dossier vide
- `rm` → permet de supprimer des fichiers

Habitudes à prendre

Sachez toujours où vous êtes !

- Si vous avez un doute : utilisez `pwd`

Sachez toujours ce qu'il y a autour de vous (dans le dossier de travail courant) !

- Si vous avez un doute : utilisez `ls`

Quand vous vous déplacez (changez de dossier de travail courant), regardez autour de vous !

- Après un `cd`, utilisez systématiquement `ls`

Quelques cas particuliers

Tous les dossiers contiennent obligatoirement deux fichiers :

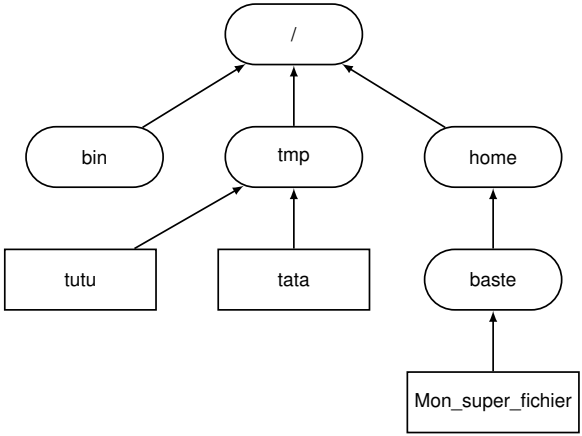
- « `.` » qui est un synonyme pour le répertoire lui-même
- « `..` » qui est un synonyme pour le répertoire qui le contient (son père)

Les fichiers dont le nom commence par un point « `.` » sont appelés fichiers cachés (par exemple par défaut la commande `ls` ne les montre pas).

Note : Sous Windows, les fichiers cachés sont gérés autrement.

- Sur une clef usb, vous pouvez avoir des fichiers caché pour Windows et visible sous Linux (et réciproquement)

Arborescence de fichiers



Le chemin relatif est le chemin allant du dossier de travail courant jusqu'au fichier ciblé.

Si je suis dans `/home/baste`
Chemin relatif pour `Mon_super_fichier` :
`Mon_super_fichier`

Si je suis dans `/home`
Chemin relatif pour `Mon_super_fichier` :
`baste/Mon_super_fichier`

Si je suis dans `/tmp`
Chemin relatif pour `Mon_super_fichier` :
`../home/baste/Mon_super_fichier`

Attention aux chemins

Chemin absolu :

- Commence par un “`/`”
- est indépendant du dossier de travail courant

Chemin relatif :

- Ne commence PAS par un “`/`”
- est relatif au dossier de travail courant
  - Implique de savoir quel est le dossier de travail courant (n’oubliez pas le `pwd`)
  - Si je change de dossier courant, le chemin relatif change aussi.
- Souvent plus pratique à utiliser car plus court

Hiérarchie standard Unix

`/bin`	Commandes utilisateurs essentielles
`/dev`	Fichiers de périphériques
`/etc`	Fichiers de configuration spécifique à la machine
`/home`	Répertoires des utilisateurs
`/lib`	Librairies partagées
`/sbin`	Commandes d'administration essentielles
`/tmp`	Fichiers temporaires
`/usr`	Seconde hiérarchie
`/var`	Données variables
`/usr/bin`	La plupart des commandes utilisateurs
`/usr/include`	Fichier d'entêtes pour les programmes C
`/usr/lib`	Librairies
`/usr/local`	Hiérarchie locale
`/usr/sbin`	Commandes d'administrations non-vitales
`/usr/share`	Données indépendantes de l'architecture
`/usr/src`	Code source

Plus de détails sur l'effort de standardisation : `http://www.pathname.com/fhs/`

C’est tout pour aujourd’hui

Un petit jeu pour ne pas vous ennuyer :

- En français : `http://luffah.xyz/bidules/Terminus/`
- En anglais : `http://www.mprat.org/Terminus/`