NUMP1 : Les commandes de base UNIX

Rémi Giraud
remi.giraud@enseirb-matmeca.fr

2026-2027

Toute question sur une fonction ? `man fonction`

1 Gestion de répertoires et fichiers

1.1 Afficher le contenu des répertoires

Avec `ls` (list) :

`ls`	(affiche les fichiers présents dans le répertoire courant)
`ls folder/`	(affiche le contenu du répertoire `folder`)
`ls toto*`	(affiche les fichiers commençant par `toto`)
`ls toto?`	(affiche les fichiers commençant par `toto` et ayant un seul caractère suivant)
`ls toto??`	(affiche les fichiers commençant par `toto` et ayant deux caractères suivants)
`ls *.txt`	(affiche les fichiers terminant par `.txt`)
`ls ?[r-u]*`	(affiche les fichiers dont la deuxième lettre est compris entre r et u)
`ls folder/*a*`	(affiche les fichiers contenant un `a` dans le répertoire `folder`)
`ls -a`	(affiche les fichiers cachés commençant par un `.`)
`ls -l`	(affiche plus d'informations, notamment les tailles et les droits)
`ls -al`	(affichage avec les deux options `-a` et `-l`)

1.2 Se déplacer

Avec `cd` (change directory) :

`cd .`	(ne bouge pas, car on se déplace dans le répertoire courant ./)
`cd folder1`	(se déplace dans le répertoire `./folder1`)
`cd folder1/`	(même chose, le `/` est facultatif ici)
`cd folder1/folder2...`	(se déplace dans plusieurs répertoires)
`cd ..`	(se déplace dans le répertoire parent)
`cd ~`	(se déplace dans le répertoire de base (`$HOME`))
`cd`	(se déplace dans le répertoire de base (`$HOME`))
`cd -`	(se déplace dans le répertoire précédent)
`cd /`	(se déplace dans le répertoire racine)
`pwd`	(affiche le chemin courant par rapport au répertoire de base (`$HOME`))

1.3 Création et copie de fichiers/répertoires

Avec `touch`, `mkdir` (make directory), `cp` (copy), `mv` (move) :

`touch file`	(crée le fichier `./file` s'il est vide, sinon aucune action)
`mkdir folder`	(crée le répertoire `./folder`)
`cp file new_file`	(copie `file` dans `new_file` (écrasé s'il existe déjà !))
`cp file folder/`	(copie `file` dans le répertoire folder et celui-ci garde le même nom)
`cp file1 file2 ... folder/`	(copie les `file1 file2 etc.` dans `folder`)
`mv ./file1 ./file2`	(renomme `file1` en `file2` (écrasé s'il existe déjà !))
`mv ./file ./folder/`	(déplace `file1` dans le dossier `folder` (`file` garde le même nom))
`mv ./file ./folder/file2`	(déplace `file1` dans le dossier `folder` et le renomme `file2`)
`mv * folder/`	(déplace tous les fichiers du répertoire courant dans `folder`)

1.4 Suppresion

Avec `rm` (remove) :

`rm file`	(supprime `file` (attention quasi définitif !))
`rmdir folder`	(supprime le dossier `folder` s'il est vide)
`rm -rf folder/`	(supprime le dossier `folder` et son contenu (faire très attention !))
`rm ./folder/*.txt`	(supprime tous les fichiers terminant par `.txt` dans `folder`)

2 Les commandes d'édition

Avec `more` (affiche le contenu d'un fichier page par page sans retour en arrière) ;
Avec `less` (affiche le contenu d'un fichier avec possibilité de se déplacer) ;
Avec `cat` (renvoie le contenu d'un fichier vers la sortie standard), etc. :

`more file`	(affiche le contenu de `file`)
`more file1 file2`	(affiche le contenu de `file1` puis `file2`)
`more *.txt`	(de tous les fichiers terminant par `.txt` dans le répertoire courant)
`cat ./folder/*.txt`	(de tous les fichiers terminant par `.txt` dans `folder`)
`less ./folder/*.txt`	(de tous les fichiers terminant par `.txt` dans `folder`)
`head file`	(affiche les 10 premières lignes de `file`)
`head -n30 file`	(affiche les 30 premières lignes de `file`)
`wc file`	(affiche le nombre de lignes, mots et octets dans le fichier `file`)
`diff file1 file2`	(affiche les différences entre les fichiers `file1` et `file2`)
`tr "[a-z]" "[A-Z]"`	(écrire : `aa bbb 11111 AAA A22` donne : `AA BBB 11111 AAA A22`)
`tr "[:lower:]" "[:upper:]"`	(écrire : `aa bbb 11111 AAA A22` donne : `AA BBB 11111 AAA A22`)

3 Liens

Avec `ln` (link) :

`ln file new_file`	(crée un lien physique entre `file` et un nouveau fichier `new_file`)
`ln -s file new_file`	(crée un lien symbolique entre `file` et un nouveau fichier `new_file`)
`ln -s folder file`	(crée un lien symbolique entre `file` et un répertoire `folder`)

4 Redirection

Avec les chevrons `>`, `>>`, `2>`, `<` :

`program > file`	(redirige le résultat du programme dans la sortie standard vers le fichier `file`)
`ls > file`	(écrit le résultat de la commande `ls` dans le fichier `file` (le crée si inexistant))
`ls -al > file`	(écrit le résultat de la commande `ls -al` dans le fichier `file` (le crée si inexistant))
`ls > file -al`	(écrit aussi le résultat de `ls -al` dans le fichier `file` (le crée si inexistant))
`ls >> file`	(écrit le résultat de la commande `ls` à la fin du fichier `file` (le crée si inexistant))
`ls -z 2> file`	(écrit le message d'erreur de `ls -z` dans le ficher `file` (le crée si inexistant))
`program < file`	(donne le contenu de `file` comme entrée standard du programme)

5 Recherche

Avec `grep` (recherche les occurrences de mots) :

`grep pattern file`	(cherche le pattern dans le fichier `file`)
`ls | grep pattern`	(cherche le pattern dans le résultat de `ls`)

Ici le *pipe* `|` se substitue à `<, >`
Par exemple `proc1 | proc2` équivaut à `proc1 > fich; proc2 < fich`

Avec `find` (recherche des fichiers) :

`find folder -type d`	(cherche tous les sous-dossiers présents dans `folder`)
`find folder -name *.a`	(les fichiers en `.a` (librairies) dans tous les sous-dossiers de `folder`)

6 Gestion des processus

Chaque programme en cours d'exécution est un processus, identifié par un numéro unique : son PID (*Process IDentifier*).

6.1 Lister les processus

Avec `ps` (process status) :

`ps`	(affiche les processus lancés depuis le terminal courant)
`ps -f`	(format complet : UID, PID, PPID (PID du parent), heure de lancement, etc.)
`ps -ef`	(affiche tous les processus du système, au format complet)
`ps aux`	(affiche tous les processus avec leur consommation CPU et mémoire)
`ps -u login`	(affiche les processus de l'utilisateur `login`)
`ps -p 2481`	(affiche le processus de PID 2481 (permet de vérifier s'il existe encore))
`ps aux | grep firefox`	(ne garde que les lignes contenant `firefox`)
`pgrep firefox`	(affiche directement le(s) PID des processus nommés `firefox`)

6.2 Surveiller les processus en temps réel

Avec `top` (affiche les processus actifs et leur consommation, rafraîchi en continu) :

`top`	(lance la surveillance en temps réel)
`top -u login`	(n'affiche que les processus de l'utilisateur `login`)

Touches utilisables pendant que `top` tourne :

`[P]`	(trie les processus par utilisation CPU (tri par défaut))
`[M]`	(trie les processus par utilisation mémoire)
`[k]`	(demande un PID puis un signal, et l'envoie au processus : équivalent de `kill`)
`[q]`	(quitte `top`)
`htop`	(version plus lisible et interactive, si installée : tri, recherche, `[F9]` pour tuer)

6.3 Terminer un processus

Avec `kill` (envoie un signal à un processus désigné par son PID) :

`kill 2481`	(demande au processus 2481 de s'arrêter proprement : signal `SIGTERM` (15))
`kill -15 2481`	(même chose, en précisant explicitement le signal)
`kill -9 2481`	(force l'arrêt immédiat du processus 2481 : signal `SIGKILL` (9))
`kill -l`	(affiche la liste des signaux disponibles)
`pkill firefox`	(envoie `SIGTERM` à tous les processus dont le nom contient `firefox`)
`pkill -9 firefox`	(force l'arrêt de tous ces processus)
`[ctrl+c]`	(interrompt le programme lancé au premier plan dans le terminal : `SIGINT`)

Attention ! Toujours essayer `kill` sans option avant `kill -9` : avec `SIGKILL`, le processus n'a pas le temps de se terminer proprement (données non sauvegardées, fichiers temporaires, ...).
On ne peut terminer que ses propres processus (sauf en tant qu'administrateur).

7 Gestion des droits

Un fichier appartient :
— à un utilisateur, qui est son propriétaire
— à un groupe (pas nécessairement celui de son propriétaire)

Le mode d'accès définit l'accès en $\begin{cases} \text{lecture (r)} \\ \text{écriture (w)} \\ \text{exécution/passage (w)} \end{cases}$ pour $\begin{cases} \text{le propriétaire (u)} \\ \text{les membres du même groupe (g)} \\ \text{les autres utilisateurs (o)} \end{cases}$.

Cela fait 9 informations élémentaires (3 champs de 3 flags), que l'on note :

$$\underbrace{\texttt{rwx}}_{u}\ \underbrace{\texttt{rwx}}_{g}\ \underbrace{\texttt{rwx}}_{o}$$

On utilise un `-` pour indiquer qu'un accès n'est pas autorisé :

`rw-r-r-`

`rwxr-xr-x`

`chmod` $< champs >^{+}_{-} < accs >$ $< chemin >$

`chmod` $< champs >=< accs >$ $< chemin >$

`chmod` $< codeoctale >$ $< chemin >$

On peut aussi représenter ce codage avec la valeur binaire associée à un code sur 3 bits. Chaque chiffre fait référence à un champ. Tout chiffre manquant est considéré comme nul. Vous pouvez convertir ce chiffre octale en 3 chiffres binaires avec des poids respectifs :

$$\underbrace{\texttt{r}}_{2^2} + \underbrace{\texttt{w}}_{2^1} + \underbrace{\texttt{x}}_{2^0} \tag{1}$$

`chmod o-r file`	(enlève le droit en lecture pour les autres utilisateurs sur `file`)
`chmod +w,o-x file`	(+ droit en écriture pour l'utilisateur, - le droit en exécution pour les autres)
`chmod g-w,o+rw file`	(- droit en écriture pour le groupe, + le droit en lecture et exé. pour les autres)
`chmod 744 file`	(Propriétaire : tous les droits ; Groupe : lecture ; Autres : lecture)
`chmod -R 777 folder`	(tous les droits à tout le monde sur `folder` et tous les sous-dossiers)

8 Dossiers spéciaux

Dossiers spéciaux depuis la racine `/` :

`/bin`	(contient les commandes de base)
`/boot`	(contient les informations nécessaires au démarrage de la machine)
`/dev`	(contient les fichiers spéciaux correspondant aux périphériques)
`/dev/null`	(périphérique nul, qui supprime toutes les données qui y sont écrites)
`/etc`	(contient la plupart des fichiers de configuration)
`/home`	(contient les répertoires personnels des utilisateurs)
`/lib`	(contient les principales bibliothèques partagées (équivalent des DLL de Windows))
`/lost+found`	(contient les fragments de fichiers perdus quand un disque crashe (réparation avec `fsck`))
`/mnt`	(contient les systèmes de fichiers montés temporairement (CD-ROM, DD, ...))
`/opt`	(c'est là qu'on installe les logiciels commerciaux)
`/proc`	(répertoire factice, contenant des infos sur l'état du système et des processus en cours)
`/root`	(le répertoire de l'administrateur système)
`/sbin`	(contient les commandes de base nécessaires à l'administration système)
`/tmp`	(contient les fichiers temporaires)
`/var`	(contient des données fréquemment réécrites)
`/usr`	(contient les logiciels installés avec le système)
`/usr/bin`	(contient les exécutables)
`/usr/include`	(contient les fichiers d'en-tête pour la programmation)
`/usr/lib`	(contient les DLL non vitales)
`/usr/local`	(contient des logiciels compilés sur place avec les sources. Organisation similaire à `/usr`)
`/usr/src`	(contient les sources de certains logiciels, principalement le noyau de Linux)

9 Variables

Pour définir une variable :

`var=valeur`	(affectation. Attention , ne pas mettre d'espace autour de =)
`$var`	(valeur de la variable `var`)
`echo $var`	(affiche la valeur de la variable `var`)
`${variable}`	(valeur de la variable (permet d'éviter certaines ambiguïtés : si `a="var"`, `${a}b` renvoie `varb` alors que `$ab` est invalide)

Variables spéciales :

`echo $?`	(affiche le statut de la dernière commande)
`echo $$`	(affiche le numéro de processus de la dernière commande)

Substitution de commandes :

`echo $(ls ./folder)`	(affiche le contenu du dossier `folder`)
`echo ‘ls ./folder‘`	(affiche le contenu du dossier `folder`)

Attention ! ici le backquote ‘ (touches `[altGR+7]`) est différent du caractère quote ’ (touche `[4]`)

10 Personnalisation de Bash

10.1 Alias

Il est possible de donner un nom particulier (`alias`) à une commande que l'on utilise fréquemment. Cette déclaration se fait dans le `.bashrc` (fichier caché) : `alias` *`nom='commande'`*

`alias ssheirb='ssh login@ssh.enseirb.fr'`	(définit l'alias (à écrire dans le `.bashrc`))
`source .bashrc`	(prend en compte les modifications faites dans `.bashrc`)
`ssheirb`	(exécute en réalité `ssh login@ssh.enseirb.fr`)
`alias`	(affiche la liste des alias connus)

10.2 Variables d'environnement

Les systèmes Unix utilisent des variables réservées pour la configuration de l'environnement. Pour modifier une variable d'environnement, utilisez la syntaxe suivante :

```
export VAR=valeur
```

La variable `PATH` donne la liste des répertoires où aller chercher les commandes. Vous pouvez voir son contenu avec la commande :

```
echo $PATH
```

Si vous voulez, par exemple, que les fichiers qui sont dans le répertoire `bin` de votre répertoire d'accueil soient directement accessibles, ajoutez le chemin dans la variable `PATH` comme suit :

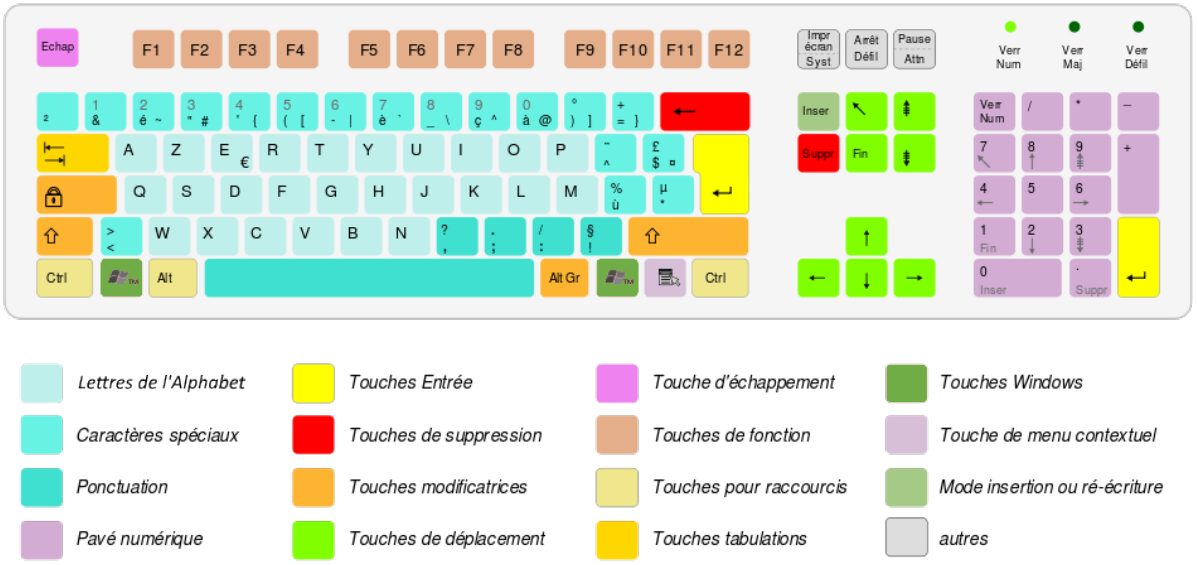
```
export PATH=$PATH:$HOME/bin
```

Vous pouvez personnaliser l'affichage du prompt (en y indiquant le chemin du dossier courant par exemple) en éditant la valeur `PS1` dans votre .profile ou `.bashrc` (ou `.bash_profile`). Un prompt secondaire Prompt String 2 (`PS2`) est utilisé lorsque la commande en cours a besoin de plusieurs lignes.

11 Raccourcis de navigation

`[tabulation]`	(complète la saisie avec ce que le terminal voit (commandes, fichiers locaux, ...))
`[double tabulation]`	(en cas d'incertitude, affiche la liste des possibilités)
`[fleche_haut]`	(remonte dans l'historique des commandes)
`[fleche_bas]`	(descend dans l'historique des commandes)
`[ctrl+fleche_gauche]`	(se déplace d'un mot vers la gauche)
`[ctrl+fleche_droite]`	(se déplace d'un mot vers la droite)
`[ctrl+a]`	(se déplace au début du prompt)
`[ctrl+e]`	(se déplace à la fin du prompt)
`[ctrl+k]`	(coupe tout à droite du curseur)
`[ctrl+y]`	(colle ce qui a été coupé)
`[sélection souris (copier) + clic_molette (coller)]`	(autre copier/coller indépendant)

12 Touches du clavier



https://www.coursinfo.fr/decouverte/souris-et-clavier/touches-dun-clavier/

13 Linux à la maison

Beaucoup de projets, TP, et enseignements de l'ENSEIRB-MATMECA s'effectuent sur des systèmes Linux. Il est donc très fortement recommandé aux élèves de configurer leur machine personnelle pour pouvoir disposer du même environnement de travail.

Plusieurs solutions existent :

— Installer une distribution Linux sans dual boot (depuis Windows 10) :
https://ubuntu.com/tutorials/ubuntu-on-windows\#1-overview

— Installer une machine virtuelle Linux :
https://thor.enseirb-matmeca.fr/ruby/docs/teaching/vmlinux

— Installer un dual boot : (à vos risques et périls)
https://www.malekal.com/installer-ubuntu-20-04-dual-boot-windows-10/
Pour les trois dernières méthodes se référer ensuite à la section Linux.

14 Se connecter à l'ENSEIRB-MATMECA depuis l'extérieur

https://yannick-bornat.enseirb-matmeca.fr/wiki/lib/exe/fetch.php?media=pg108:acces_em_distance.pdf