TP1–2 · Commandes et scripts shell

NUnix · L3 MIAGE · Université de Rennes

Elisabeth Bouriah et Louis Ledoux 2026–2027

Utiliser une machine Linux avec Bash.

1. Pour commencer

Se connecter à son compte Linux et ouvrir un terminal.

1. Lancer `pwd`, puis `echo "$HOME"`. Que montre chaque commande ? Si le terminal démarre dans votre répertoire personnel, pourquoi les résultats sont-ils identiques ? Quelle commande permet d'y revenir depuis n'importe où ?
2. Créer un répertoire `NUnix` dans votre répertoire personnel avec `mkdir`. Quelle commande permet de vérifier sa création ?
3. Y entrer avec `cd`. Essayer de compléter son nom avec **Tab**. Si plusieurs noms commencent de la même façon, appuyer deux fois sur Tab pour les afficher.
4. Parcourir les commandes précédentes avec les flèches **Haut** et **Bas**. Afficher à nouveau le chemin absolu du répertoire courant.
5. Essayer le copier-coller dans le terminal. Dans de nombreux terminaux Linux, sélectionner du texte puis cliquer sur le bouton central de la souris colle la sélection. Vous pouvez aussi utiliser **Ctrl+Shift+C** et **Ctrl+Shift+V**.
6. Créer un répertoire `tp1` dans `NUnix`. Ce sera le répertoire de travail du TP.
7. Télécharger tp-files.zip soit avec le navigateur (enregistrer dans `~/NUnix`), soit depuis un terminal dans `~/NUnix` avec `wget URL` (remplacer `URL` par l'adresse du lien). Dans `~/NUnix`, lancer `unzip tp-files.zip`. Vérifier que `files` et `tp1` sont deux répertoires distincts dans `NUnix`.

2. La commande man

`man commande` affiche le manuel d'une commande. Utiliser **Espace** pour avancer d'une page, **Entrée** pour avancer d'une ligne et **q** pour quitter.

Consulter les commandes et options suivantes. Pour les commandes internes à Bash, comme `cd` et `history`, utiliser `help cd` ou `help history`.

```
man, man -k
cd, cd .., ls, ls -a, ls -l, ls -al, pwd
mkdir, rmdir, cp, cp -R, mv, rm, rm -R
touch, echo, history, history -c, who, passwd, cat
```

3. Variables d'environnement

Stocker le chemin des fichiers fournis dans une variable pour y accéder depuis un autre répertoire :

```
export nom_variable=chemin_repertoire
```

1. Entrer dans `~/NUnix/files` et trouver son **chemin absolu** avec `pwd`. Stocker ce chemin dans une variable d'environnement nommée `tpfiles`.
2. Revenir dans `~/NUnix`. Utiliser `tpfiles` pour lister le contenu de `files`.
3. L'utiliser pour afficher le contenu de `prog1.c`.

4. Redirections

1. Copier le contenu du répertoire désigné par `tpfiles` dans `~/NUnix/tp1` avec `cp`. Travailler dans `~/NUnix/tp1` pour la suite des exercices.
2. Ouvrir `prog1.c` dans un éditeur de texte. Que fait ce programme ? Lors du lancement d'un éditeur graphique depuis le terminal, essayer d'ajouter `&` pour l'exécuter en arrière-plan et continuer à utiliser le shell.
3. Lancer `make` dans `~/NUnix/tp1` pour compiler les programmes fournis (un compilateur C est nécessaire). Puis exécuter `./prog1` dans chacun des cas suivants :

Entrée standard	Sortie standard
Clavier	Écran
Fichier `d1`	Écran
Clavier	Fichier `res1`
Fichier `d1`	Fichier `res2`

   Lors d'une saisie au clavier, terminer par `%`, le caractère d'arrêt du programme.

5. Scripts shell

1. Lire `prog2.c`. Son entrée standard contient un entier `i`, suivi d'une suite de caractères séparée en parties par `%` :
   ```
   premiere partie%deuxieme partie%troisieme partie%
   ```
   Il écrit la `i`ème partie sur la sortie standard. Les parties sont numérotées à partir de 1.
2. Exécuter `./prog2` avec plusieurs entrées pour comprendre son fonctionnement.
3. Écrire un script `select.sh`, appelé avec trois arguments :
   ```
   ./select.sh i f1 f2
   ```
   L'entier `i` désigne la partie à extraire, `f1` est le fichier contenant la suite de caractères et `f2` est le fichier de sortie. Utiliser `prog2` pour extraire la partie demandée. Le fichier `d2` fourni peut servir d'entrée.

   Vérifier le nombre d'arguments, l'existence des fichiers et les droits d'accès. Ces formes de `cat` peuvent vous aider :
   ```
   cat f1 f2
   cat - f
   ```
   La première concatène deux fichiers. La seconde lit l'entrée standard, puis le fichier `f`.
4. Écrire `selmult`, appelé sous la forme `./selmult f`. Il lit en boucle un entier `i` et affiche la `i`ème partie du fichier `f` en utilisant `prog2`. Arrêter lorsque l'utilisateur saisit `0`.

6. Extraction des informations système

`ps` affiche les processus en cours ; `ps -A` sélectionne tous les processus. `find` recherche dans le système de fichiers. Une redirection telle que `2>/dev/null` élimine les messages d'erreur.

1. Créer `aliveprocs.txt` contenant les noms de tous les processus en cours (colonne du nom de commande). Afficher le fichier.
2. Lister les processus triés par taille de mémoire virtuelle. Afficher le résultat. Consulter `man ps` pour les champs disponibles et leurs unités.
3. Lister les processus qui n'appartiennent pas à `root`, triés par nom.
4. Lister les noms des processus dans l'ordre de leur temps CPU total. Si un même nom apparaît plusieurs fois, ne l'afficher qu'une fois.
5. Afficher les noms des fichiers ordinaires de votre répertoire personnel qui ont été modifiés aujourd'hui.
6. Créer `read-only.txt` contenant les chemins des fichiers ordinaires du système appartenant à l'utilisateur qui exécute la commande.

7. Programme d'édition

Écrire un programme Bash qui recherche une chaîne `str` dans les fichiers d'un répertoire donné et remplace ses occurrences par une autre chaîne, `new_str`.

1. Accepter des options avec une valeur, comme `./main.sh -a valeur`.
2. Utiliser des fonctions dans le programme.
3. Prévoir une option d'aide expliquant les arguments et leur utilité.
4. Vérifier les arguments et retourner un code d'erreur s'ils sont invalides.
5. Ajouter ensuite le choix entre remplacer la première occurrence et remplacer toutes les occurrences.

Par exemple, `-h` peut afficher l'aide et `-p` désigner le répertoire. Choisir et documenter les autres options.

Utiliser une arborescence comme celle-ci pour essayer le programme sur des fichiers situés à différentes profondeurs :

```
.
├── file.txt
├── folder1/
│   ├── file.txt
│   └── sub1/
│       ├── file.txt
│       └── subsub1/
│           └── file.txt
├── folder8/
│   └── file.txt
└── main.sh
```