# Initiation CLI sous GNU/Linux

(ou comment se passer du clicodrome en utilisant la console…)

► *à lire pour devenir une bête de la console* ◄

## I. Utilisation de quelques commandes de base du Shell Bash pour créer, supprimer, déplacer répertoires et fichiers et se déplacer dans l'arborescence

1. Vérifiez que le répertoire courant est bien votre répertoire de travail avec la commande `pwd` (et **n'en bougez plus** pour la suite : commande `cd` interdite et utilisation de chemins absolus interdite !)
2. Créez un répertoire `tpconsole`
3. Dans `tpconsole`, créez deux répertoires `rep1` et `rep2`
4. Dans `rep1`, créez un fichier texte ascii `myname` contenant votre nom et votre prénom (commande `echo` et redirection). Les accents sont autorisés ! (codage par défaut : Unicode utf-8)
5. Copiez ce fichier dans `rep2` (sans changer le nom du fichier).
6. Renommer le fichier de `rep2` en `monnom`
7. Affichez la taille de `monnom` et ses droits (permissions d'accès) avec la commande `ls -l`
8. Modifiez les droits d'accès de ce fichier pour qu'il soit accessible en écriture par tout le monde (`chmod` avec deux méthodes)
9. Afficher son contenu (commande `cat`)
10. Utilisez la commande `tree` pour visualiser l'arborescence de `tpconsole`
11. Supprimez `myname` (vérifiez le résultat avec `tree tpconsole`)
12. Supprimez `tpconsole` et tout ce qu'il contient

## II. Exemple de programmation en mode console : *Shell Bash*

1. Avec `nano`, éditez le script suivant (enregistrez-le dans un fichier `double.sh`) :

```
#!/bin/bash
read -p'Entrez un nombre entier : ' n
echo "Le double de $n est $((2*$n))"
```

2. Vérifier que ce fichier n'a pas encore les droits d'exécution
3. Lui donner les droits d'exécution (`chmod a+x double.sh`)
4. Exécutez-le (`./double.sh`)

## III. Exemple de programmation en mode console : *langage C*

1. Avec `nano`, éditez le programme suivant (enregistrez-le dans un fichier `double.c`) :

```
#include <stdio.h>
int main()
{
 unsigned int n;
 printf ("Entrez un nombre entier : ");
 scanf ("%d", &n);
 printf ("Le double de %d est %d\n", n, 2*n);
 return 0;
}
```

2. Compilez ce programme (`gcc -Wall -o double double.c`)
3. Vérifiez que le fichier `double` créé a bien les droits d'exécution
4. Exécutez-le (`./double`)

## IV. Exemple de programmation en mode console : *Python 3*

1. Avec `nano`, éditez le programme suivant (enregistrez-le dans un fichier `double.py`) :

```
#!/usr/bin/python3
n = int(input("Entrez un nombre entier : "))
print("Le double de", n, "est", 2*n)
```

2. Exécutez-le : `python3 double.py`
3. Vérifier que ce fichier n'a pas encore les droits d'exécution
4. Lui donner les droits d'exécution (`chmod a+x double.py`)
5. Exécutez-le (`./double.py`)

## V. Quelques commandes utiles : `xxd`, `file`, `top`, `wget`...

1. Essayez par exemple : `xxd double.sh`
2. Dans un répertoire contenant des fichiers divers, essayez : `file *`
3. `top` (ou mieux `htop`) : voir les processus
4. Télécharger un fichier dans le répertoire courant : `wget http://fdec.fr/fd.png`

## VI. Enchaînement de commandes

1. Essayez `rm fichierquinexistepas || date`
2. puis `rm fichierquinexistepas && date`
3. puis `rm fichierquinexistepas ; date`

Recommencez en inversant les deux commandes.

## VII. Tubes (en anglais « pipelines »)

1. Essayez la commande `ps -eo pmem,pcpu,pid,args`
2. puis `ps -eo pmem,pcpu,pid,args | tail -n +2`
3. puis `ps -eo pmem,pcpu,pid,args | tail -n +2 | sort -rnk 1`
4. puis `ps -eo pmem,pcpu,pid,args | tail -n +2 | sort -rnk 1 | head -n 5`
5. puis `ps -eo pmem,pcpu,pid,args | head -n 1 && ps -eo pmem,pcpu,pid,args | tail -n +2 | sort -rnk 1 | head -n 5`

Grâce à `ls`, `grep`, `sort` et `head`, afficher les 4 plus gros fichiers du répertoire `/usr/bin` dont le nom contient la chaîne de caractères `python`

## VIII. Expansions Bash

1. En vous aidant de https://www.youtube.com/watch?v=82ESpisUh3Q, concevez une commande permettant de créer huit fichiers (utilisez la commande `touch`) vides nommés `fichier001.txt fichier010.txt fichier011.txt … fichier111.txt`.

2. Complétez pour obtenir en retour la chaîne de caractères `defgh` :

```
var='abcdefghij'; echo ${ à compléter }
```

3. Complétez pour obtenir en retour la chaîne de caractères (on veut remplacer les chiffres par des croisillons)
`le produit de ## par ## est ###:`

```
var='le produit de 12 par 13 est 156'; echo ${ à compléter }
```

4. Testez la commande `echo "il est $(date +%H) heures."`
puis créez vous-même une commande permettant d'illustrer la « substitution de commande » de type `$(commande)`.