

Cultures numériques avancées

Exécution des scripts en ligne de commande.

Ljudmila PETKOVIC

ljudmila.petkovic@sorbonne-nouvelle.fr

```
~/Documents/linux-commands via v3.9.6
> ls -lah
Permissions Size User Date Modified Name
drwxr-xr-x - daniel 8 ago 15:11 .
drwxr-xr-x - daniel 8 ago 00:27 ..
drwxr-xr-x - daniel 8 ago 00:34 commands
drwxr-xr-x - daniel 7 ago 00:45 dir1
drwxr-xr-x - daniel 7 ago 00:45 dir2
drwxr-xr-x - daniel 8 ago 00:10 dir_to_copy
drwxr-xr-x - daniel 8 ago 00:12 new_dir
-rw-r--r-- 0 daniel 8 ago 00:38 BestMoviesOfAllTime
-rw-r--r-- 0 daniel 7 ago 00:44 binarysearch.py
-rw-r--r-- 0 daniel 7 ago 00:43 dummyfile1.txt
-rw-r--r-- 0 daniel 8 ago 00:18 file_to_delete.txt
-rw-r--r-- 0 daniel 7 ago 00:44 get_keys.py
-rw-r--r-- 0 daniel 7 ago 00:44 github_automation.py
-rw-r--r-- 0 daniel 7 ago 00:44 important_file.txt
-rw-r--r-- 0 daniel 8 ago 00:04 new_file.txt
-rw-r--r-- 0 daniel 12 abr 20:45 old_file
```

Source : [Diaz, 2023](#).

Cultures numériques avancées (L2HN001)

Mineure « Humanités numériques », licence Lettres

Paris, le 9 février 2024, année 2023-2024

Diapositives adaptées de [Simon Gabay](#), [Simone Rebora](#) et [Élisabeth Brunet et Gaël Thomas](#).

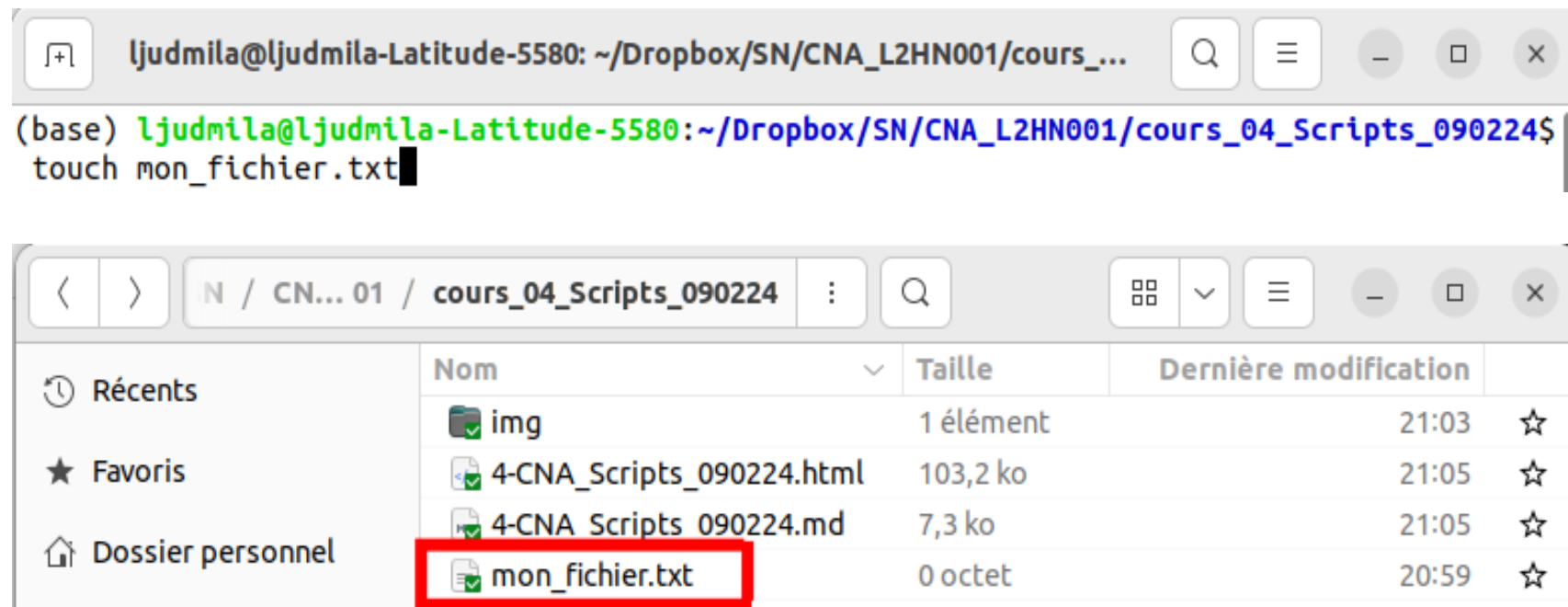
Créer un fichier en ligne de commande

- il existe plusieurs manières de créer un fichier en ligne de commande
- alternative aux éditeurs de texte basés sur une interface graphique
 - Notepad, TextEdit, Gedit...

touch

- commande `touch` suivi du nom du fichier que l'on veut créer :

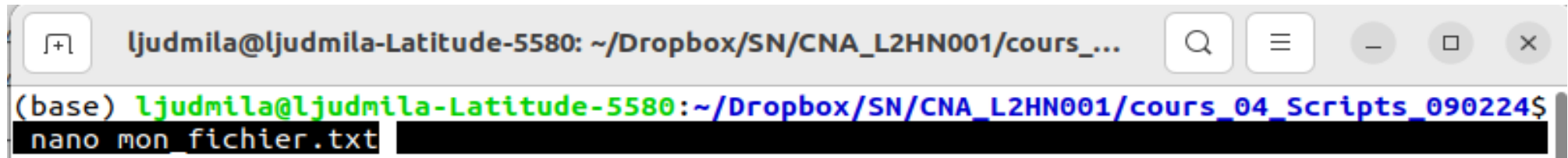
```
touch mon_fichier.txt
```



nano

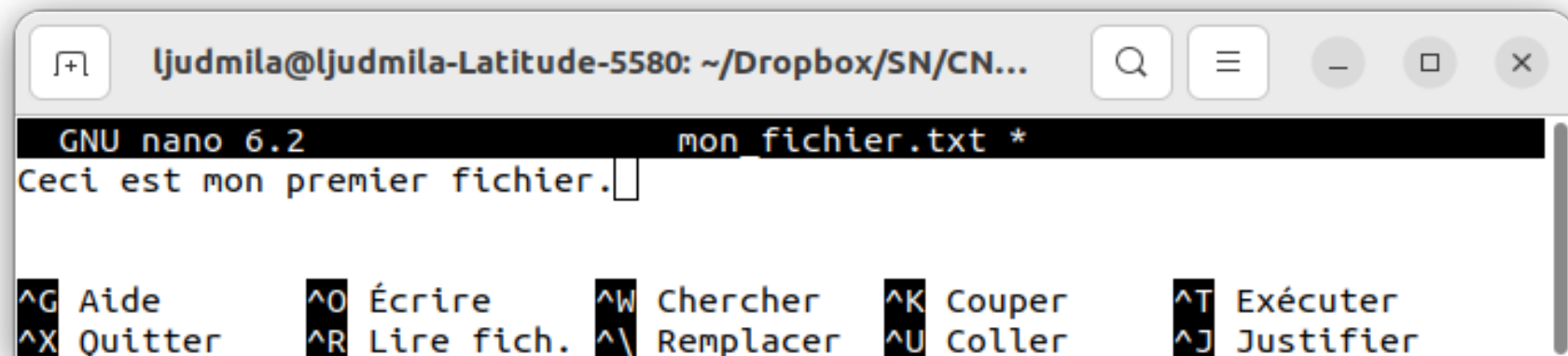
- éditer les fichiers en ligne de commande avec l'éditeur nano

```
nano mon_fichier.txt
```



A terminal window with a title bar showing the user 'ljudmila' at host 'ljudmila-Latitude-5580' in the directory '~/Dropbox/SN/CNA_L2HN001/cours_...'. The prompt is '(base)'. The command 'nano mon_fichier.txt' has been entered and is highlighted in black.

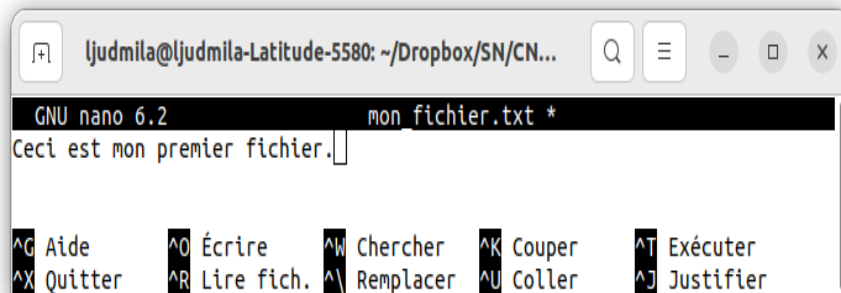
- la fenêtre ci-contre apparaîtra, où nous pouvons écrire du texte :



The nano text editor interface is shown. The title bar indicates the user 'ljudmila' at host 'ljudmila-Latitude-5580' in the directory '~/Dropbox/SN/CN...'. The editor title is 'GNU nano 6.2' and the file being edited is 'mon_fichier.txt *'. The text 'Ceci est mon premier fichier.' is written in the editor. At the bottom, a help menu lists various commands: ^G Aide, ^O Écrire, ^W Chercher, ^K Couper, ^T Exécuter, ^X Quitter, ^R Lire fich., ^\ Remplacer, ^U Coller, and ^J Justifier.

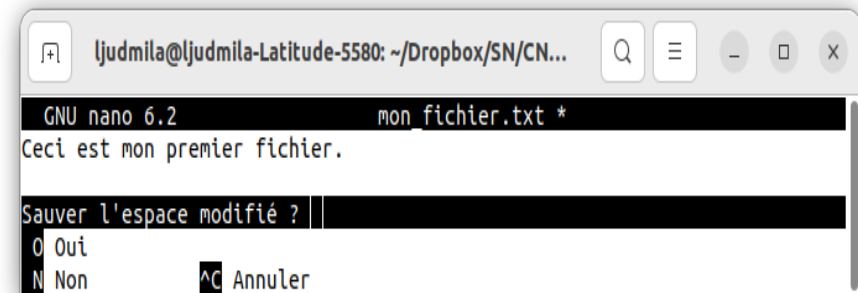
nano

- pour sortir et enregistrer le fichier, taper successivement `Ctrl + X` puis `o` (oui)
- pour finir, valider avec la touche Entrée
- les instructions sont toujours affichées en bas de l'écran



The screenshot shows the nano text editor interface. The title bar indicates the user is ljudmila@ljudmila-Latitude-5580 and the file path is ~/Dropbox/SN/CN... The editor is editing a file named 'mon_fichier.txt'. The content of the file is 'Ceci est mon premier fichier.' The bottom status bar shows various keyboard shortcuts: ^G Aide, ^O Écrire, ^W Chercher, ^K Couper, ^T Exécuter, ^X Quitter, ^R Lire fich., ^\ Remplacer, ^U Coller, and ^J Justifier.

Sortir : `Ctrl + X`



The screenshot shows the nano text editor interface, similar to the previous one. The title bar and file name are the same. The content of the file is 'Ceci est mon premier fichier.' The bottom status bar shows the prompt 'Sauver l'espace modifié ?' with two options: 'O Oui' and 'N Non'. The ^C Annuler shortcut is also visible.

Enregistrer le fichier : `o` , Entrée

Exercice 1 : créer un fichier `.txt`

1. créer un nouveau fichier `test.txt`
2. y écrire le contenu suivant : *Bonjour le monde.*
3. sortir de l'éditeur de texte et enregistrer le fichier
4. vérifier la présence du fichier dans le répertoire courant
 - i. utiliser la commande déjà abordée aux cours précédents

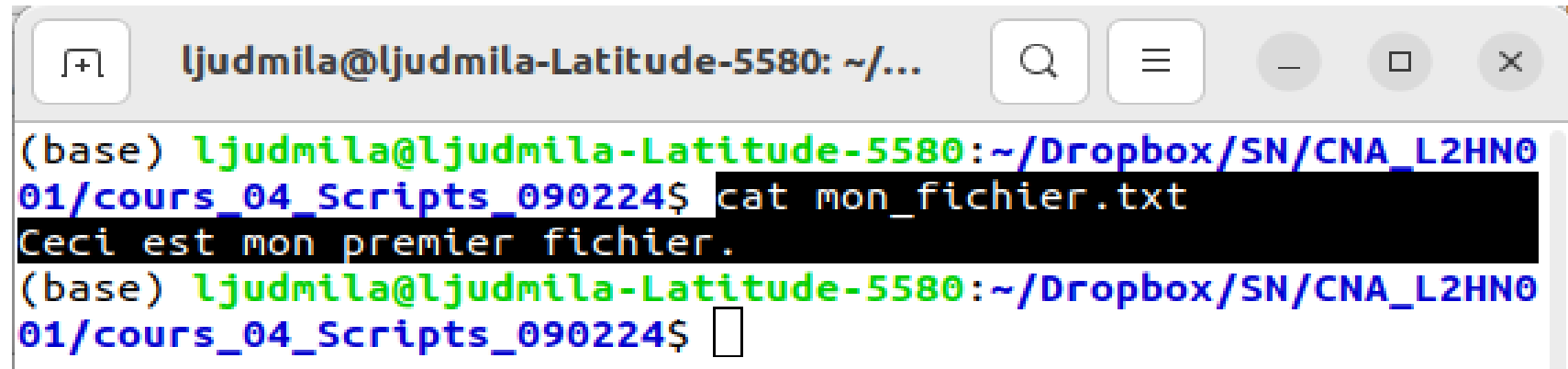
Solution de l'exercice 1

1. `nano test.txt`
2. saisie du texte *Bonjour le monde.*
3. `Ctrl + X`, `O`
4. `ls`

Lecture du fichier : `cat`

- la commande `cat` permet de lire le contenu du fichier

```
cat mon_fichier.txt
```

A terminal window with a title bar showing the user 'ljudmila' on a machine named 'ljudmila-Latitude-5580'. The prompt is '(base)'. The user has entered the command 'cat mon_fichier.txt' and the output is 'Ceci est mon premier fichier.'.

```
ljudmila@ljudmila-Latitude-5580: ~/...  
(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN01/cours_04_Scripts_090224$ cat mon_fichier.txt  
Ceci est mon premier fichier.  
(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN01/cours_04_Scripts_090224$
```


Redirection du contenu : > et >>

- créer le fichier `source.txt` et y écrire le message « Ceci est un autre fichier.
- pour rediriger le contenu du fichier `source` vers le fichier `destination`, utiliser l'option `>`

```
cat source.txt > destination.txt
```

- pour **ajouter** le contenu dans le fichier de destination, utiliser l'option `>>`

```
cat source.txt >> destination.txt
```

Le `>` est l'opérateur de redirection de sortie utilisé pour **écraser** les fichiers qui existent déjà dans le répertoire. Bien que le `>>` soit également un opérateur de sortie, il **ajoute** les données d'un fichier existant.

Recherche du texte : `grep`

- pour faire des recherches du texte dans un fichier, utiliser la commande `grep` :

```
grep "Ceci" source.txt
```

- on peut faire des requêtes en utilisant les expressions régulières (*regex*).
 - trouver tous les mots commençant par la lettre « C » en majuscule

```
grep -Eo "C\w+" source.txt
```

`-E` : expression rationnelle étendue (≠ expression rationnelle simple `-G`)

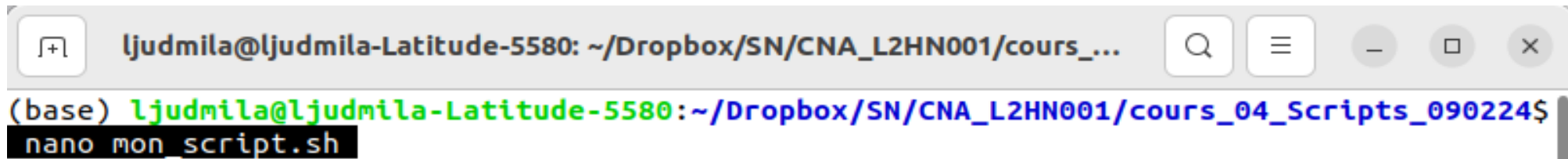
`-o` : n'afficher que l'occurrence en question (*match*)

`C` : le caractère littéral « C »

`\w+` : un ou plusieurs caractères alphanumériques

Créer un script

- on peut créer des scripts *shell* pour automatiser certaines tâches
 - traitement des données, la manipulation de fichiers, etc.
- on crée un script shell avec la commande `nano mon_script.sh`
 - on utilise l'extension `.sh` pour nommer les scripts (fichier *shell*)

A screenshot of a terminal window. The title bar shows the user 'ljudmila' on a machine named 'ljudmila-Latitude-5580' in the directory '~/Dropbox/SN/CNA_L2HN001/cours_...'. The terminal content shows the prompt '(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts_090224\$' followed by the command 'nano mon_script.sh' which is highlighted in black background with white text.

```
(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts_090224$ nano mon_script.sh
```

Créer un script

- on déclare le type de document avec un appel de script (***shebang***)

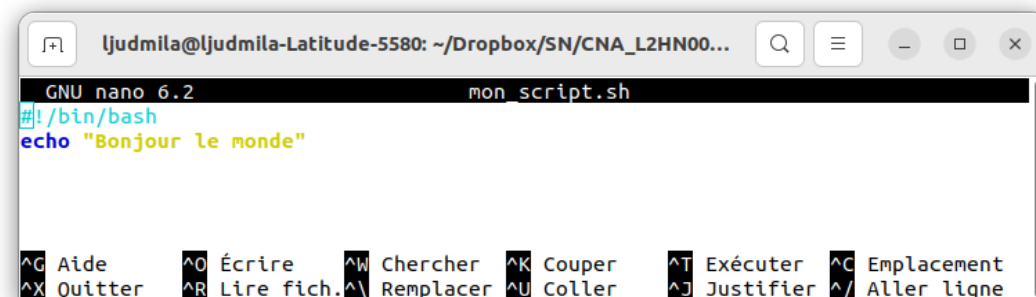
```
#!/bin/bash
```

- cela indique que le fichier n'est pas un fichier binaire mais un script

- on écrit le texte suivant dans l'éditeur de texte `nano` :

```
echo "Bonjour le monde"
```

- on sort de l'éditeur et on enregistre le fichier (cf. p. 5)

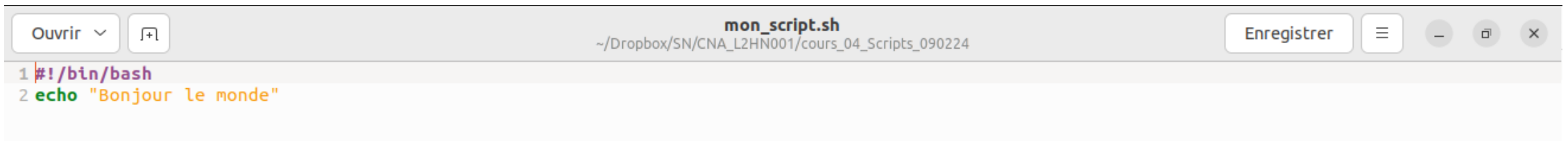


```
ljudmila@ljudmila-Latitude-5580: ~/Dropbox/SN/CNA_L2HN00...
GNU nano 6.2 mon_script.sh
#!/bin/bash
echo "Bonjour le monde"

^G Aide    ^O Écrire  ^W Chercher ^K Couper  ^T Exécuter ^C Emplacement
^X Quitter ^R Lire fich. ^M Remplacer ^U Coller  ^J Justifier ^_ Aller ligne
```

Fichier créé

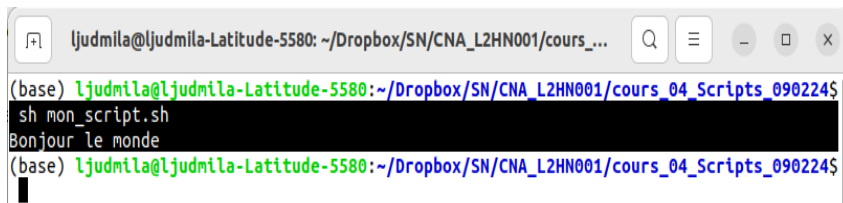
- le fichier `mon_script.sh` est enregistré dans le répertoire courant
- quand on l'ouvre, on voit qu'il contient du texte saisi lors de l'étape précédente
 - commande qui permet d'afficher le texte « Bonjour le monde »

A screenshot of a code editor window. The title bar shows the filename 'mon_script.sh' and the path '~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts_090224'. On the left, there is a button 'Ouvrir' with a dropdown arrow and a file icon. On the right, there is a button 'Enregistrer', a hamburger menu icon, and standard window control buttons (minimize, maximize, close). The editor area contains two lines of code: '1 #!/bin/bash' and '2 echo "Bonjour le monde"'.

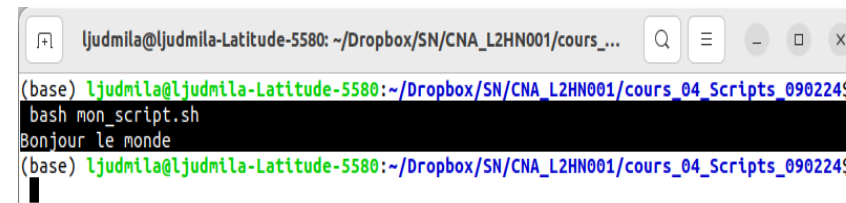
```
1 #!/bin/bash
2 echo "Bonjour le monde"
```

Exécution du script

- avant de lancer le script, le `mon_script.sh` doit être **exécutable**
 - pour lancer le script, taper `sh mon_script.sh` ou `bash mon_script.sh`
 - le message « Bonjour le monde » s'affiche

A terminal window screenshot showing a user at the prompt (base) ljudmila@ljudmila-Latitude-5580: ~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts_090224\$ typing 'sh mon_script.sh'. The output is 'Bonjour le monde' followed by a new prompt (base) ljudmila@ljudmila-Latitude-5580: ~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts_090224\$.

`sh mon_script.sh`

A terminal window screenshot showing a user at the prompt (base) ljudmila@ljudmila-Latitude-5580: ~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts_090224\$ typing 'bash mon_script.sh'. The output is 'Bonjour le monde' followed by a new prompt (base) ljudmila@ljudmila-Latitude-5580: ~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts_090224\$.

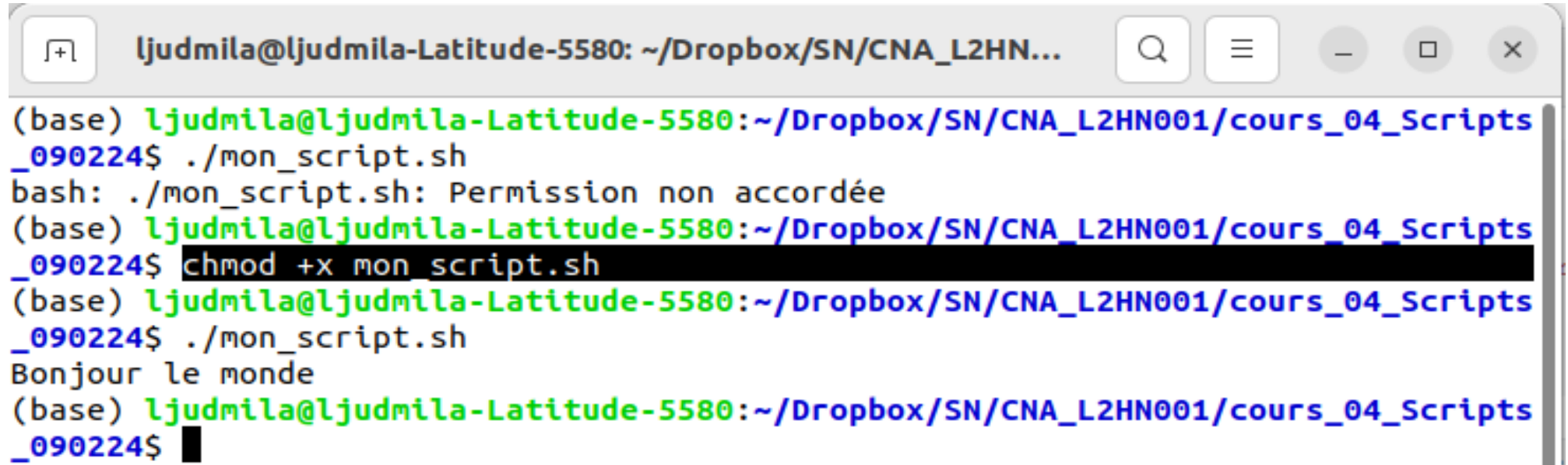
`bash mon_script.sh`

Exécution du script (Windows)

- autre manière d'exécuter le script :

```
./mon_script.sh
```

- si vous obtenez le message d'erreur : `bash: ./mon_script.sh: Permission non accordée`, il faut rendre le fichier exécutable en tapant `chmod +x mon_script.sh` et puis relancer le script `./mon_script.sh`



```
ljudmila@ljudmila-Latitude-5580: ~/Dropbox/SN/CNA_L2HN...  
(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts  
_090224$ ./mon_script.sh  
bash: ./mon_script.sh: Permission non accordée  
(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts  
_090224$ chmod +x mon_script.sh  
(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts  
_090224$ ./mon_script.sh  
Bonjour le monde  
(base) ljudmila@ljudmila-Latitude-5580:~/Dropbox/SN/CNA_L2HN001/cours_04_Scripts  
_090224$
```

Exercice 2 : créer un script `.sh`

1. créer un script `list.sh`
2. y écrire le *shebang*, puis la commande `ls`
3. sortir de l'éditeur de texte et enregistrer le fichier
4. vérifier la présence du fichier dans le répertoire courant
5. lancer le script et expliquer le résultat
 - i. si besoin, rendre le fichier exécutable
6. créer une copie de ce script (`list_copie.sh`) dans le répertoire courant
7. supprimer la copie créée

Solution de l'exercice 2

1. `nano list.sh`
ou `touch list.sh`, puis `nano list.sh`
2. `#!/bin/bash`
`ls`
3. `Ctrl + X`, puis `O`
4. `ls`
5. `sh list.sh`
ou `bash list.sh`
ou `chmod +x list.sh`, puis `./list.sh`
Ce script affiche le contenu du répertoire courant.
6. `cp list.sh list_copie.sh`
7. `rm list_copie.sh`

Commentaires

- un commentaire sert à améliorer la lisibilité du script
- il est placé en le faisant précéder du caractère **dièse** (ou croisillon, « # »)
- tout ce qui suit la dièse est ignoré (non exécuté) jusqu'à la fin de la ligne
- cela permet de mettre un commentaire sur une ligne d'instructions
- l'espace sépare la fin de l'instruction et le début du commentaire

```
#!/bin/bash  
# Ceci est un commentaire dans mon script shell qui ne va pas être exécuté.  
ls # Cette ligne indique la commande qui affiche le contenu du répertoire courant.
```

Exercice 3 : créer un script avec un commentaire

1. créer un script `date.sh`
2. y écrire
 - i. le *shebang*
 - ii. le commentaire « Ce programme affiche la date »
 - iii. la commande `date` suivi du commentaire « Cette ligne affiche la date »
3. sortir de l'éditeur de texte et sauvegarder le script créé
4. lancer le script et expliquer le résultat

Solution de l'exercice 3

1. `nano date.sh`

ou `touch date.sh`, puis `nano date.sh`

2.

```
#!/bin/bash
# Ce programme affiche la date
date # Cette ligne affiche la date
```

3. `Ctrl + X`, puis `o`

4. `sh date.sh`

`bash date.sh`

`chmod +x date.sh`, puis `./date.sh`

Ce script affiche et définit la date et l'heure du système.

Variables

Le shell permet de définir et réutiliser des variables :

```
message="qui suis-je ?"  
echo $message  
qui suis-je ?
```

Il ne faut **pas** d'espace autour du signe `=`.

Les guillemets sont nécessaires si la valeur de la variable contient des espaces.

Le méta-caractère dollar (`$`), suivi d'un nom de variable (`message`), est remplacé par la valeur de la variable en question (`qui suis-je ?`).

(Champin, 2020)

Variables : méta-caractère `$`

- le méta-caractère `$` n'est pas inhibé par les guillemets doubles (`"`)

```
$ echo "J'ai une question : $message"  
J'ai une question : qui suis-je ?
```

- en revanche, il l'est par les guillemets simples (`'`)

```
$ echo 'J'ai une question $message'  
> # la commande ne s'exécute pas
```

Adapté de [Champin \(2020\)](#)

Exercice 4 : créer un script contenant une variable

1. créer un script `coucou.sh`
2. y écrire le *shebang*
3. affecter la valeur `Michel` à la variable `nom`
4. formuler la commande d'affichage du message « Coucou [nom] »
 - i. on récupère la valeur de `nom` à partir de la variable déclarée à l'étape 3.
5. ajouter le commentaire « Cette ligne affiche le nom 'Michel' »
6. sortir de l'éditeur de texte et enregistrer le script
7. lancer le script et expliquer le résultat

Solution de l'exercice 4

1. `nano coucou.sh`

ou `touch coucou.sh`, puis `nano coucou.sh`

2. `#!/bin/bash`

3. `nom="Michel"`

4. `echo "Coucou $nom"`

5. `# Cette ligne affiche le nom 'Michel'`

6. `Ctrl + X`, puis `o`

7. `sh coucou.sh`

`bash coucou.sh`

`chmod+x coucou.sh`, puis `./coucou.sh`

Affecter le résultat d'une commande à une variable

Introduisons une nouvelle variable `date_actuelle`.

Pour stocker le résultat d'une commande dans cette variable :

- mettre la commande entre parenthèses, précédées par un `$`

```
date_actuelle=$(date)
```

- ou mettre la commande entre les accents graves ``

```
date_actuelle=`date`
```

Exercice 5 : créer un script contenant une variable II

1. créer un script `variable.sh`
2. ajouter le *shebang*
3. stocker la commande `date` dans la variable `date_actuelle`
4. formuler la commande d'affichage du message « Aujourd'hui on est `date` »
 - i. on récupère la valeur de `date` à partir de la variable déclarée à l'étape 3.
5. ajouter le commentaire « Cette ligne affiche la date »
6. sortir de l'éditeur du texte et enregistrer le script
7. lancer le script et expliquer le résultat

Solution de l'exercice 5

1. `nano variable.sh`
ou `touch variable.sh`, puis `nano variable.sh`
2. `#!/bin/bash`
3. `date_actuelle=$(date)`
4. `echo "Aujourd'hui on est $date_actuelle"`
5. `# Cette ligne affiche la date`
6. `Ctrl + X`, puis `0`
7. `sh variable.sh`
`bash variable.sh`
`chmod +x variable.sh`, puis `./variable.sh`

Avantages de l'utilisation de la ligne de commande

- **Flexibilité** : combiner les commandes et obtenir une palette pratiquement infinie de fonctions nouvelles
- **Fiabilité** : tendance à s'exécuter de la même manière sur différents SE (« couteau suisse »)
- **Rapidité** : automatisation des tâches à grande échelle (p. ex. renommer un ensemble des fichiers d'un seul coup)
- **Expérience** : communiquer avec l'ordinateur plus directement qu'avec les interfaces graphiques, en apprenant ainsi énormément sur son fonctionnement interne
- **Économisation des ressources** : utilise les ressources de l'ordinateur beaucoup plus parcimonieusement que les programmes graphiques

Source : [Floss Manuals](#)

Références

- **Brunet, É. & Thomas, G.** (s. d.). « Le shell bash » [*diapositives*]. Télécom SudParis. <http://www-inf.telecom-sudparis.eu/cours/CSC3102/Supports/ci1-bash/ci-bash.pptx.pdf>
- **Combeau, M.** (2022). « La différence entre le terminal, la console et le shell ». <https://www.codequoi.com/difference-entre-terminal-console-et-shell/>
- **Champin, P.-A.** (2020). Séance 5 : Automatisation et scripts. Département Informatique (IUT Lyon 1). <https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/linux/s5.html>
- **Rebora, S.** (2022). « Connaître son propre ordinateur » [*dépôt GitHub*]. EnExDi2022. https://github.com/ABC-DH/EnExDi2022/tree/main/materials/1_KnowYourComputer/slides
- **Gabay, S.** (2022). « Les lignes de commandes et Bash » [*dépôt GitHub*]. Université de Genève, Chaire des humanités numériques, Faculté des Lettres. https://github.com/gabays/Fondamentaux/blob/main/Lignes_de_commandes/DistRead_1_2.pdf
- **Diaz, D.** (2023). « Les 40 commandes Linux les plus utilisées que vous devez connaître ». Kinsta. <https://kinsta.com/fr/blog/commandes-linux/>
- **Xyoo** (s.d.). « Interface graphique ». <https://cours-informatique-gratuit.fr/dictionnaire/interface-graphique/>