Pratique des machines

TP2 : Sortie standard, redirections, connexions ssh et manuel

am@up8.edu

Septembre 2025

- **Ne pas copier / coller d'instructions depuis le PDF** : cela ajoute des espaces partout qui rendent les commandes incorrectes
- Il n'y a rien à rendre. Par contre, je vous encourage à prendre des notes et **si vous ne savez pas répondre à une des questions du TP ou que vous ne comprenez pas le résultat d'une commande, vous devez poser la question**

Table des matières

1 Préparez votre espace de travail **2**

2 Sortie standard et redirections **3**

2.1 Créer un fichier vide **3**

2.1.1 Créer un fichier sans l'ouvrir 3

2.1.2 Créer un fichier en l'ouvrant avec un éditeur de texte 4

2.2 redirection en sortie **6**

3 Connexion aux machines depuis l'extérieur **6**

4 Envoyer des fichiers en passant par ssh **7**

5 Pour la semaine prochaine : apprendre à utiliser le manuel 8

Manier le terminal

Quelques astuces et raccourcis clavier pour utiliser le terminal :

- `Ctrl + Maj + t` : ouvrir un nouvel onglet dans le terminal
- (sur certains systèmes) `Ctrl + Alt + t` : ouvrir un nouveau terminal
- signe & en fin de commande permet de garder l'accès au terminal
- `Ctrl + d` : arrêter le processus en cours ou fermer le terminal (ex : lancez la commande `cat` puis arrêtez le processus)
- la tabulation permet d'auto-compléter
- flèches du haut / bas : se déplacer dans l'`historique` des commandes lancées
- `Ctrl + a` : aller au début de la ligne
- `Ctrl + e` : aller à la fin de la ligne
- `Ctrl + k` : couper jusqu'à la fin de la ligne
- `Ctrl + Maj + C` : clic droit "copier" : copier la région en surbrillance
- `Ctrl + Maj + V` : clic droit "coller" ou clic molette ou `Ctrl + y` (seulement au sein du terminal) : coller
- `Ctrl + c` : arrêter la commande en cours
- `clear` ou `Ctrl + l` : effacer les commandes précédentes de l'interface graphique
- `reset` : réinitialiser l'environnement du shell

1 Préparez votre espace de travail

Dans un dossier qui pourrait être `~/Documents/Cours/pdm` (vous avez le choix dans l'organisation des fichiers sur votre machine), créez un dossier `TP2` avec la commande `mkdir` et faites en sorte que ce dossier devienne votre répertoire de travail grâce à la commande `cd`.

2 Sortie standard et redirections

On appelle `entrée standard` le flux d'entrée par lequel du texte ou toute autre donnée peut être entré dans un programme, et `sortie standard` flux de sortie dans lequel les données sont écrites par le programme.

Dans le terminal, le flux d'entrée (les commandes) sont saisies au clavier dans l'`invite de commande`. Le flux en sortie s'affiche sur l'écran dans la fenêtre du terminal. C'est le cas par exemple du résultat de la commande `ls`.

Pour plus de commodité dans le traitement du résultat d'une commande, il est possible de `rediriger le flux de sortie` `vers un fichier`, c'est à dire qu'au lieu de s'afficher sur l'écran, le résultat de la commande sera écrit dans un fichier.

Il est aussi possible de `rediriger le flux de sortie` `vers une autre commande` pour enchaîner les commandes progressivement.

Les opérateurs suivants permettent de rediriger la sortie standard vers :

- un fichier en écrasant son contenu : avec l'opérande "`chevron` droit" `>`
- un fichier en concaténant la sortie au contenu existant avec "double chevron droit" `>` `>`
- une nouvelle commande avec l'opérande "`pipe`" : `|`

2.1 Créer un fichier vide

2.1.1 Créer un fichier sans l'ouvrir

→ ouvrez le manuel de la commande `touch`, et utilisez-la pour créer un fichier vide appelé `fichier_touch.txt`.
→ vérifiez que le fichier a bien été créé avec la commande `ls`.

```
$ touch [OPTION]... FILE...
```

touch - change file timestamps
Cette commande change la date de dernière modification du fichier dont le chemin est passé en argument. C'est utile notamment lorsqu'on souhaite recompiler des fichiers même s'ils n'ont pas été modifiés depuis la dernière compilation. Lorsque le chemin

donné en argument n'est pas un fichier, alors le fichier est créé.

→ Identifiez où cette information est donnée dans la documentation suivante.

```
SYNOPSIS
       touch [OPTION]... FILE...

DESCRIPTION
       Update  the  access  and modification times of each FILE to the current
       time.

       A FILE argument that does not exist is created empty, unless -c  or  -h
       is supplied.

       A  FILE  argument  string of - is handled specially and causes touch to
       change the times of the file associated with standard output.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a     change only the access time

       -c, --no-create
              do not create any files

       -d, --date=STRING
              parse STRING and use it instead of current time

       -f     (ignored)

       -h, --no-dereference
              affect each symbolic link instead of any referenced file (useful
              only on systems that can change the timestamps of a symlink)

       -m     change only the modification time

       -r, --reference=FILE
              use this file's times instead of current time

       -t STAMP
              use [[CC]YY]MMDDhhmm[.ss] instead of current time

       --time=WORD
              change the specified time: WORD is access, atime, or use: equiv-
              alent to -a WORD is modify or mtime: equivalent to -m

       --help display this help and exit

       --version
              output version information and exit

       Note that the -d and -t options accept different time-date formats.
```

2.1.2 Créer un fichier en l'ouvrant avec un éditeur de texte

Il existe de nombreux éditeur de textes, avec ou sans `interface graphique`.

— sans interface graphique : vim, nano, emacs, etc.
— avec interface graphique : gedit, kate, vscode, emacs GUI, etc.

→ Mais d'ailleurs, que veut dire "GUI" ?

Ils peuvent tous être utilisés pour ouvrir ou créer un fichier avec la commande :

```
$ [text-editor] [path-to-file]
```

La commande `editor` ouvre un éditeur de textes par défaut.

→ trouvez l'éditeur de texte par défaut installé sur votre machine.
→ créez un nouveau fichier `fichier_editeur.txt`, ajoutez une ligne dedans puis sauvegardez-le et fermez-le.

Attention : programme GUI et utilisation de &

Lorsque vous utilisez une commande qui ouvre un programme muni d'une interface graphique (GUI), vous devez utiliser l'opérande `&` qui permet de lancer le programme `en toile de fond`.

Explication : lorsque vous lancez une commande, vous ne pouvez récupérer l'accès à votre prompt qu'une fois que celle-ci est terminée. Or, si vous lancez un programme avec une interface graphique, la commande ne sera considérée comme terminée que lorsque vous aurez fermé le programme. Entre temps, votre terminal sera inutilisable car vous n'avez plus accès au prompt.

→ essayez d'exécuter la commande suivante :

```
$ while true; do echo "coucou" ; sleep 2 ; done
```

→ Que se passe-t-il ? → Exécutez `Ctrl-C` dans le terminal. Vous venez d'interrompre la commande lancée ci-dessus. → essayez d'ouvrir dans le terminal un éditeur de texte avec GUI comme :

```
$ gedit
```

L'éditeur s'ouvre, mais si vous revenez sur le terminal, vous n'avez plus accès au prompt.

→ Exécutez `Ctrl-C` dans le terminal. La fenêtre s'est fermée car vous avez interrompu le pro-

gramme.

Pour pouvoir conserver l'accès à votre terminal, vous devez donc utiliser l'opérande `&` qui indique à votre shell de lancer la commande dans un autre shell (invisible pour vous).
→ Par exemple :

```
$ gedit fichier_gedit.txt &
```

2.2 redirection en sortie

→ testez les commandes suivantes et expliquer ce qu'il se passe. À quoi sert la commande `cat` ?

```
$ echo "ligne 1"
$ echo "ligne 1" > sortie.txt
$ cat sortie.txt
$ echo "ligne 1" >> sortie.txt
$ cat sortie.txt
$ echo "ligne 2" >> sortie.txt
$ cat sortie.txt
$ echo "ligne 3" > sortie.txt
$ cat sortie.txt
```

3 Connexion aux machines depuis l'extérieur

Vous pouvez vous connecter à votre session "bocal" depuis une machine à l'extérieur du réseau de l'université. La commande utilisée est `ssh`.

$ ssh [OPTION]... LOGIN@MACHINE [COMMAND [ARGUMENT...]]

secure shell : **ssh** (client SSH)
Ouvre une `session ssh` sur une `machine distante` afin de pouvoir exécuter des commandes sur celle-ci. L'argument `COMMAND` est optionnel.

- en l'absence de l'argument `COMMAND`, ssh se connecte et ouvre une session sur la `MACHINE` de destination (avec éventuellement un nom d' utilisateur `LOGIN`. Les Login autorisés sont ceux pour lesquels il existe un compte sur la machine de destination.
- Si `COMMAND` est spécifiée, alors la commande est exécutée sur la machine distante.

→ Si vous avez une machine personnelle avec vous, testez de vous connecter en ssh sur le compte de votre voisin·e (vous prenez son login) sur l'infrastructure du bocal avec :

```
$ ssh login@bocal.cs.univ-paris8.fr
```

avec :

- `LOGIN` = login de votre voisin·e
- `MACHINE` = nom du serveur du Bocal

Comme vous vous connectez sur le compte de votre voisin·e c'est à lui/elle de taper son mot de passe pour vous laisser accès.

Vous pouvez arrêter la session ssh en tapant :

```
$ exit
```

→ Réessayez en ajoutant la commande `pwd` qui vous indique quel est le répertoire de travail par défaut lorsque vous vous connectez à une machine distante.

```
$ ssh login@bocal.cs.univ-paris8.fr pwd
```

→ pour la semaine prochaine : connectez vous depuis chez vous (ou votre machine personnelle) à votre espace personnel du bocal.

4 Envoyer des fichiers en passant par ssh

Vous pouvez transférer des fichiers de manière sécurisée d'une machine à l'autre. On utilise pour cela la commande `scp`.

$ scp [OPTION]... SOURCE... TARGET

secure copy : scp est un programme qui permet de copier un fichier (`SOURCE`) sur une autre machine (`DESTINATION`).
Comme pour ssh, il faudra spécifier le mot de passe de la machine destination.

→ Si vous avez une machine personnelle sur vous, créez un fichier `votrenom.txt` sur votre machine, et copiez le sur la machine de votre voisin·e (login : xxx) avec :

```
$ scp votrenom.txt xxx@bocal.cs.univ-paris8.fr:~
```

Si tout s'est bien passé, votre voisin·e devrait maintenant avoir un fichier à votre nom dans son *home*.

La commande peut bien sûr être utilisée pour transférer des fichiers (ou archives contenant plusieurs fichiers) depuis la machine du bocal vers votre machine personnelle en inversant les valeurs de `SOURCE` et de `DESTINATION`.

5 Pour la semaine prochaine : apprendre à utiliser le manuel

$ man [OPTIONS] [[SECTION] PAGE ...] ...

man - Interface de consultation des **man**uels de référence du système

Pour avoir de l'aide sur une commande, il est possible de consulter son manuel d'utilisation avec la commande `man` (pour « manual »), par exemple :

```
$ man cp
```

vous donne le manuel de la commande correspondant à la `copie de fichier`.

→ À quoi servent les commandes suivantes ?
- `wc`
- `sort`
- `uniq`

→ Combien d'arguments prennent les commandes suivantes ?
- `pwd`
- `find`
- `cp`
- `grep`

→ Quelles sont les options à utiliser
- pour afficher sur la sortie standard un texte dont les lignes ont été classées par ordre décroissant (`sort`)
- pour afficher sur la sortie standard un texte dont les lignes ont été classées aléatoirement (`sort`)
- pour lister les fichiers du dossier courant en affichant leurs tailles (`ls`)

— pour lister les fichiers du dossier courant en affichant leurs tailles de manière « lisible par un humain » (`ls`)