T5.2 OS et commandes UNIX

{: .center width=640}

5.2.1 Système d'exploitation

Le système d'exploitation (OS) est le logiciel système qui gère l'ordinateur, en jouant un rôle d'intermédiaire entre les programmes, les éléments matériels de l'ordinateur (notamment la mémoire) ainsi que les périphériques d'entrée/sortie. C'est le premier programme exécuté au démarrage de la machine, et c'est le seul qui reste en permanence en exécution.

{: .center}

La partie de l'OS qui gère la communication avec les éléments matériels s'appelle **le noyau** (kernel).

Outre le noyau, l'OS comporte des applications, une interface graphique, la gestion des fichiers, le terminal...

Historiquement, les premiers systèmes d'exploitation ne disposaient pas d'interface graphique. D'ailleurs, à cette époque, la souris n'existait même pas. On interagissait donc avec le système essentiellement par échange de texte. L'écran servait à recevoir les infor- mations fournies par le système sous forme de phrase, d'affichage de valeur ou de tableau textuel. Aujourd'hui, même si les interfaces graphiques modernes permettent d'effectuer la plupart des opérations, il est important de connaitre quelques-unes de ces lignes de commandes qui se saisissent dans un (émulateur de) terminal, qui lance un shell par défaut.

!!! info "Vocabulaire" === "Terminal" {: .center}

```
    L'**émulateur de terminal** (souvent appelé **terminal** tout court) est un programme qu

    Par défaut le terminal exécute un **shell**.

    Il existe de nombreux émulateurs de terminaux sous Unix: xterm, rxvt-unicode, LXTerminal
=== "Shell"

    Le shell (aussi appelé invite de commande ou interpréteur de commandes) est un programme

    Il existe de nombreux shell sous Unix. Les plus utilisés dans le monde Linux sont bash e

=== "Commande"

    Une commande est en général un petit programme externe lancé par le shell. Par exemple,
```

Principalement, les commandes servent à lancer des petits programmes ou à travailler sur les fichiers et les répertoires. Dans les systèmes de type "UNIX" (par exemple GNU/Linux ou macOS), nous avons un système de fichier en arborescence :

{: .center}

5.2.2 Commandes

Pour découvrir un certain nombre de commandes UNIX, parmi les plus courantes, nous allons jouer à un jeu:

{: .center}{:target="_blank"}

!!! example “Travail à faire” En jouant à Terminus:

```
- noter au fur et à mesure les commandes découvertes, à quoi elles servent et comment on les

| Nom de la commande | Description | Utilisation |
|--------------------|-------------|-------------|
| `ls`               | Lister les éléments du dossier | Saisir `ls` puis *Entrée*|

- faire un plan du jeu sous forme d'une arborescence, par exemple:

![](../images/terminus_plan_debut.png){: .center}
```

5.2.3 Compléments sur les commandes

On reprend l’exemple de l’arborescence du dossier `/home/jeanmichel/` de l’ordinateur de Jean-Michel du DS 6:

```
.
+-- Travail
|   +-- reforme
|   +-- HTML
|   |   +-- images
|   |   |   +-- toto.png
|   |   |   +-- tata.png
|   |   |   +-- tutu.png
|   |   +-- styles
|   |   |   +-- style.css
|   |   +-- scripts
|   |   |   +-- script.js
|   |   +-- index.html
|   |   +-- page1.html
|   |   +-- page2.html
+-- Photos
|   +-- Ibiza
|   |   +-- reveillon.png
```

!!! abstract “Chemins” On a vu la commande `cd` qui permet de changer de répertoire (dossier). Par exemple, pour aller dans le répertoire `images` depuis la racine,

on peut taper successivement: bash ~$ cd Travail/ ~/Travail$ cd HTML/ ~/Travail/HTML$ cd images/ ~/Travail/HTML/images$

On peut plus rapidement préciser le chemin:
```bash
~$ cd Travail/HTML/images/
~/Travail/HTML/images$
```

Pour revenir à la racine `/home/jeanmichel/`, on peut taper successivement 3 fois la commanc
```bash
~/Travail/HTML/images$ cd ../../../
~$
```

Et pour aller du répertoire `images` au répertoire `scripts`, on tapera:
```bash
~/Travail/HTML/images$ cd ../scripts/
~/Travail/HTML/scripts$
```

!!! abstract “Options et arguments” === “Options” On peut ajouter aux commandes UNIX une ou plusieurs options, qui s’écrivent soit:

    - avec une lettre précédée d'un tiret;
    - avec un mot-clé précédé de 2 tirets.

    === "Exemple avec `ls`"
        Tester par exemple dans un de vos répertoires les commandes suivantes:
        ```bash
        $ ls
        $ ls -l
        $ ls -l -h
        ```
    === "Exemple avec `rm`"
        Attention, l'usage de `rm` est dangereux, soyez attentif !

        La commande `rm` permet de supprimer **définitivement** un fichier, il n'y a pas de

        Pour effacer tous les fichiers d'un répertoire:
        ```bash
        $ rm *
        ```

        Et pour supprimer tout le contenu d'un répertoire, y compris les sous-répertoires, c
        ```bash
        $ rm -r *
        ```

```
=== "Arguments"
    On peut parfois préciser plusieurs arguments à une commande pour ne pas répéter plusieur
    ```bash
    $ mkdir images/ data/
    ```
    Et trois fichiers `toto.txt`, `tata.txt` et `tutu.txt`:
    ```bash
    $ touch toto.txt tata.txt tutu.txt
    ```
```

!!! abstract "À propos de `grep`" Comme vu dans l'activité «Terminus», la commance `grep` permet de rechercher une chaîne de caractéres dans un fichier. La syntaxe générale est: `bash $ grep options "recherche" chemin` Voici quelques options utiles:

```
- `--ignore-case` ou `-i` : pour ignorer la casse (minuscules/majuscules indifférentes);
- `-c` : pour afficher seulement le nombre d'occurences de la recherche;
- `-l` : pour afficher le nom des fichiers contenant la recherche (`chemin` est alors un rép
- `-r` : pour rechercher dans tous les fichiers et sous-répertoires de `chemin`, qui est un

Pour rechercher dans plusieurs fichiers (comme dans l'activité Terminus) on peut utiliser `*
```bash
$ grep 'password' *.txt
```
```

5.2.4 Utilisateurs et droits

Un système UNIX est un système multi-utilisateur. Toute personne physique ou tout programme interagissant avec le système est un **utilisateur** (user). Cet utilisateur est authentifié sur le système par un nom unique et un identifiant unique (UID). Chaque utilisateur possède certains droits lui permettant d'effectuer certaines opérations et pas d'autres (avoir accès aux répertoires et fichiers, aux périphériques, installer des logiciels...).

Pour connaître les utilisateurs de votre système, on consulte le fichier `/etc/passwd` (faites-le) où on trouve tous les utilisateurs au format:

`nom:motdepasse:UID:GID:informations:repertoire:shell`

Chaque utilisateur appartient à un ou plusieurs groupes, qui servent à ressembler plusieurs utilisateurs pour leur attribuer des droits (permissions d'accès) communs aux fichiers ou applications.

Pour connaître les utilisateurs de votre système, on consulte le fichier `/etc/group` (faites-le).

{: .center width=50%}

Parmi les utilisateurs, il y a un **super-utilisateur** appelé `root` qui a tous les pouvoirs sur le système. Son UID est 0. Pour exécuter une commande réservée au super-utilisateur, un utilisateur doit utiliser la commande `sudo` (super user do) qui nécessite de connaître le **mot de passe root** qui en général n'est connu que de l'administrateur du système.

En particulier le super-utilisateur peut modifier les droits (en attribuer ou en retirer) des utilisateurs et des groupes.

!!! abstract "Les types de droits `r-w-x`"

```
- les droits en **lecture** (symbolisés par la lettre `r`) : il est possible de lire le cont
- les droits en **écriture** (symbolisés par la lettre `w`) : il est  possible de modifier l
- les droits en **exécution** (symbolisés par la lettre `x`) : il est possible d'exécuter le
```

!!! abstract "Les types d'utilisateurs `u-g-o`" Tout fichier UNIX:

```
- possède un propriétaire (par défaut l'utilisateur qui l'a créé) : **u** comme *user*;
- est associé à un groupe dont on définit les actions sur ce fichier: **g** comme *group*;
- peut être éventuellement manipulé par tous les autres utilisateurs : **o** comme *others*.
```

!!! example "Lecture des droits" Voici ce que me donne la commande `ls` avec l'option `-l` pour obtenir des informations sur le contenu du répertoire `/Travail/1NSI/Archi/`:

```
![](../images/capture_terminal.png){: .center}

- Le premier caractère `-` ou `d` indique s'il s'agit d'un fichier ou d'un répertoire;
- les 9 caractères suivants représentent dans l'ordre les droits pour les 3 types d'utilisat
- ensuite on lit le nombre de liens (notion non étudiée cette année);
- on trouve ensuite le nom du propriétaire du fichier, le nom du groupe, la taille du fichie

![](../images/permissions.png){: .center width=50%}
```

!!! example "Modification des droits" Il est important de ne pas perdre de vu que l'utilisateur "root" a la possibilité de modifier les droits de tous les utilisateurs.

```
Le propriétaire d'un fichier peut également modifier les permissions d'un fichier ou d'un ré

Elle s'utilise ainsi, en précisant l'utilisateur (**a** pour tous), l'ajout ***** ou la supp

```bash
chmod [u g o a] [+ - =] [r w x] nom_du_fichier
```
Par exemple,
```bash
chmod g+w toto.txt
```
attribuera la permission "écriture" au groupe associé au fichier `toto.txt`.
```

5.2.5 Exercices

{{ initexo(0) }}

!!! example "{{ exercice() }}" === "Énoncé partie 1" Décrire (papier/crayon) l'effet de chacune des commandes suivantes en supposant qu'elle sont exécutées les unes à la suite des autres.

    1. `cd ~`
    2. `mkdir T5`
    3. `mkdir T5/TP_shell`
    4. `cd T5/TP_shell`
    6. `touch toto.txt`
    7. `ls -l`
    8. `chmod g-rwx,o-rwx toto.txt`
    9. `ls -l`
    9. `cd ..`
    10. `rm -r TP_shell`

```
=== "Énoncé partie 2"
    Ouvrir un terminal et effectuer ces commandes. Vérifier que tout se déroule comme décrit

=== "Correction"
    {{ correction(False,
    "
    "
    ) }}
```

!!! example "{{ exercice() }}" === "Énoncé" On suppose que l'on se trouve dans un répertoire vide `TEST/` et qu'on exécute les commandes suivantes. Dessiner l'arborescence finale des fichiers et répertoires en utilisant `TEST/` comme racine de l'arborescence.

    1. `mkdir series films mangas musique`
    2. `touch films/sw.txt mangas/HxH.txt`
    3. `cd series/`
    4. `mkdir ../musique/rap/ got/ oitnb/`
    5. `cd ..`
    6. `cp */*.txt series/got/`
    7. `rm -r mangas`

```
=== "Correction"
    {{ correction(False,
    "
    "
    ) }}
```

!!! example "{{ exercice() }}" === "Énoncé" Consulter la page de manuel

d'utilisation de la commande `head`. Trouver comment l'utiliser pour n'afficher que les 5 premières lignes d'un fichier `toto.txt`.

```
=== "Correction"
    {{ correction(False,
    "
    ```bash
    $ head -n 5 toto.txt
    ```
    "
) }}
```

!!! example “TP noté” === “Énoncé” {{ correction(False, " Commencer par télécharger ce fichier dans votre répertoire `/home/nsi/Téléchargements/`.

```
    Les instructions suivantes doivent **toutes** être réalisées à l'aide d'une (ou plusieu

    1. Consultez le répertoire courant (dans lequel vous vous situez). Si ce n'est pas votre
    2. Créez un répertoire `Terminal/` dans le répertoire `Documents/`, puis un répertoire `
    3. Déplacez le fichier `foo.txt` que vous avez téléchargé vers le répertoire `TP/` que v
    4. Renommez ce fichier en `queneau1014poemes.txt`.
    5. Si le répertoire courant n'est pas `/home/nsi/Documents/Terminal/TP/`, s'y rendre. Sa
    6. Créez une copie du fichier `queneau1014poemes.txt` dans le répertoire `Sauvegarde/` s
    5. Dans le répertoire `TP/`, consultez les 14 premières lignes du `queneau1014poemes.txt
    6. Parmi les 3 mots suivants, lequel(s) n'y figure(nt) pas (peu importe la casse)? Malab
    7. Donnez le nombre d'occurences du mot «frère».
    8. En quelle ligne trouve-t-on le mot «illustre»?
    9. On peut écrire du texte (par exemple «Bonjour») dans un fichier (par exemple «toto.tx
    10. Vérifiez en affichant le contenu du fichier `reponses.txt`.
    11. Consultez le contenu du répertoire `TP/`, en affichant les détails sur les droits.
    12. Modifiez les droits sur le fichier `reponses.txt` pour que les utilisateurs du group
    13. Supprimez le répertoire `Terminal/` et tout son contenu.
    14. Fermez le terminal.

    Vous pouvez maintenant aller sur Moodle. Dans le dépôt «TP shell» de la section «Évaluat
    "
    ) }}
```