TP/Cours – Système d'exploitation

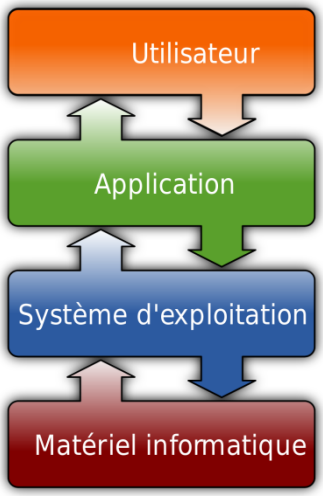
Objectifs :

- ⇨ Définir ce qu'est un système d'exploitation et comprendre son rôle
- ⇨ Manipuler les commandes de base du shell dans un système d'exploitation libre

I – Qu'est-ce qu'un système d'exploitation ?

1) Définition

Le **système d'exploitation** ou OS (**O**perating **S**ystem) est le logiciel qui gère l'utilisation des ressources matérielles de l'ordinateur et les interactions avec les logiciels applicatifs et l'utilisateur.



D'après wikipédia

Il permet donc :

- du point de vue utilisateur de :
  - Gérer le matériel
  - Exécuter les autres programmes
  - Fournir une interface homme-machine
- du point de vue logiciel de :
  - Fournir des services basiques : allouer de la mémoire, lire un fichier, etc.
  - Abstraire le matériel (= fournir des services communs pour tous les matériels)
  - Garantir la sécurité de l'ordinateur

2) Principales fonctions du système d'exploitation

Un système d'exploitation est constitué de différents éléments :

a. Le noyau

C'est le cœur du système. Il est lui-même constitué de plusieurs sous-parties comme les systèmes de fichiers, l'ordonnanceur, le gestionnaire de mémoire, … Il offre les services suivants :

- Gestion de l'accès aux fichiers : Les applications peuvent demander l'accès aux fichiers et le système d'exploitation résout les conflits d'accès simultanés, les droits et la sécurité tout en offrant un certain niveau d'abstraction (l'application n'a pas besoin de connaître précisément le format ou le lieu de stockage des fichiers).
- Gestion de l'accès aux ressources : permet aux applications de se partager les ressources matérielles en mettant en place des files d'attentes par exemple (comme les *spoolers* d'impression) et en gérant les droits d'accès et la sécurité.
- Gestion de la mémoire : C'est le système d'exploitation qui gère l'intégralité de la mémoire et l'affecte aux applications en fonction de leurs demandes.
- Gestion des applications : Le système d'exploitation permet aux applications de fonctionner simultanément en leur allouant du temps d'exécution sur le processeur à tour de rôle (c'est l'ordonnanceur qui réparti le temps d'exécution entre les différents programmes). Il démarre et termine également les applications.
- Gestion des erreurs : Signale et limite l'impact des erreurs dans les applications ou les défaillances sur le matériel et les périphériques.
- Accès au réseau

b. L'invite de commande

C'est un moyen d'accéder directement à certaines fonctions du système d'exploitation ou à sa configuration. Il existe plusieurs variantes de l'invite de commande (ou *Shell* en anglais), comme bash, ksh, zsh, ...

L'invite de commande fonctionne uniquement en mode texte. Il existe un mode interactif, où on tape les commandes les unes à la suite des autres et un mode script où on peut créer de vrais programmes qui exécutent des commandes du système d'exploitation.

c. L'interface graphique

L'interface graphique est une IHM (**I**nterface **H**omme **M**achine) permettant d'interagir avec le système d'exploitation et les applications dans un environnement graphique.

3) Différents systèmes d'exploitation

Les premiers ordinateurs dans les années d'après-guerre ne possédaient pas de systèmes d'exploitation et leurs programmes adressaient directement le matériel sans passer par un intermédiaire. La puissance des ordinateurs commençant à vraiment décoller dans les années 1960, on a vu apparaître des embryons de système d'exploitation qui géraient l'exécution des programmes via des systèmes de file d'attente.
Puis en 1965 le MIT crée Multics, le premier système d'exploitation multitâche et multiutilisateurs.
Puis UNIX voit le jour en 1969 et il faut attendre l'arrivée des premiers ordinateurs personnels (les « micro »-ordinateurs) à partir de 1980 pour qu'apparaissent de nombreux systèmes d'exploitation alternatifs (MS-DOS en 1980, MacOS en 1984, Amiga OS et la première version de Windows en 1985, OS/2 en 1987, …
Windows commence à devenir l'acteur majoritaire sur le marché des PC à partir de sa version 3 en 1990, tandis que la première alternative libre arrive avec Linux en 1991.
Puis les différentes versions de windows le rapprochent petit à petit un peu plus des OS « professionnels » de type UNIX ou VMS, tandis qu'apparaissent des systèmes d'exploitation dédiés aux matériels embarqués comme les premiers Pocket PC en 1999 ou les téléphones mobiles avec IOS et Android en 2007.

4) Système d'exploitation « libre »

Le système UNIX est un système dit "propriétaire" (certaines personnes disent "privateur"), c'est-à-dire un système non libre. Mais plus généralement, qu'est-ce qu'un logiciel libre ?
D'après Wikipédia : "Un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification et la duplication par autrui en vue de sa diffusion sont permises, techniquement et légalement, ceci afin de garantir certaines libertés induites, dont le contrôle du programme par l'utilisateur et la possibilité de partage entre individus".
Le système UNIX ne respecte pas ces droits (par exemple le code source d'UNIX n'est pas disponible, l'étude d'UNIX est donc impossible), UNIX est donc un système "propriétaire" (le contraire de "libre"). Attention qui dit logiciel libre ne veut pas forcément dire logiciel gratuit (même si c'est souvent le cas), la confusion entre "libre" et "gratuit" vient de l'anglais puisque "free" veut à la fois dire "libre", mais aussi gratuit.

Tux le manchot, la mascotte du noyau Linux

En 1991, un étudiant finlandais, Linus Torvalds, décide de créer un clone libre d'UNIX en ne partant de rien (on dit "from scratch" en anglais) puisque le code source d'UNIX n'est pas public. Ce clone d'UNIX va s'appeler Linux (Linus+UNIX).

Depuis, le système Linux a connu de nombreuses déclinaisons (on parle de « distributions ») comme Debian, Ubuntu, Red Hat ou SUSE. Ces distributions sont basées sur le même noyau, mais sont fournies avec des ensembles de logiciels, shell et interfaces graphiques différents. Certaines sont généralistes, tandis que d'autres sont spécialisées pour tel ou tel emploi comme Edubuntu (orienté vers l'éducation), Kali (orienté sécurité réseau), raspbian (pour ordinateur Raspberry Pi), Tails (vie privé & anonyma), Free-EOS (pour les serveurs), …

Dans le TP qui suit, nous utiliserons la distribution grand public « ubuntu ».

5) Processus de démarrage

Lorsque l'ordinateur démarre, le processeur commence par exécuter le programme situé en mémoire morte et qui contient les fonctions de base pour manipuler le matériel : c'est le BIOS[1] (**B**asic **I**nput **O**utput **S**ystem).
Après un inventaire et une vérification sommaire du matériel disponible sur l'ordinateur (mémoire, clavier, souris, écran, cartes graphiques, disques durs, ports usb, …), le BIOS va chercher sur le disque principal le programme de démarrage du système d'exploitation (bootloader) et l'exécute. Ce programme va charger le noyau du système et lui passer la main.

Le BIOS n'est donc *pas* un système d'exploitation. C'est un programme qui réside en permanence en mémoire morte, qui est exécuté au démarrage de la machine et dont certaines fonctions sont appelées par encore par la suite par le système d'exploitation pour gérer le matériel.

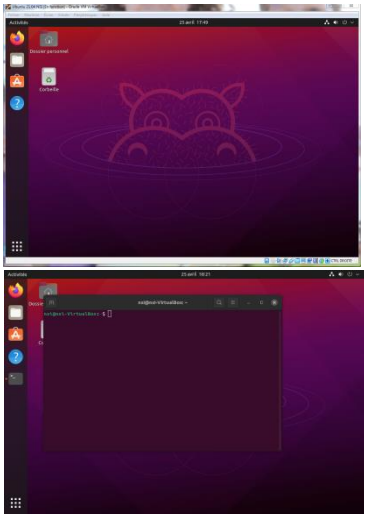
Une description plus détaillée du processus de boot peut être trouvée sur https://www.malekal.com/processus-demarrage-uefi-windows/.

II – Mise en pratique

Dans cette partie, nous allons mettre en pratique nos connaissances sur la ligne de commande dans un environnement linux.
Pour ce faire, nous utiliserons la solution de virtualisation Virtualbox sur PC/Windows.
Virtualbox est un logiciel de virtualisation, c'est-à-dire qu'il permet de simuler le fonctionnement d'un autre ordinateur dans une fenêtre. Bien qu'il puisse faire tourner tout type de système d'exploitation, nous l'utiliserons pour faire fonctionner une distribution ubuntu (distribution linux la plus répandue).

Logo de Virtualbox

1) Mise en place et démarrage de la machine virtuelle



Lancer Virtualbox, sélectionner la machine virtuelle ubuntu et la démarrer.
Le lancement prend alors un peu de temps car c'est vraiment tout le démarrage de la machine qui a lieu.
Une fois sur l'écran du bureau (voir image ci-contre), cliquer sur les 9 points blancs en bas à gauche et taper « terminal ».

On ouvre alors une invite de commande qui va nous permettre de répondre aux questions du TP.

2) Arborescence, chemins relatifs et absolus

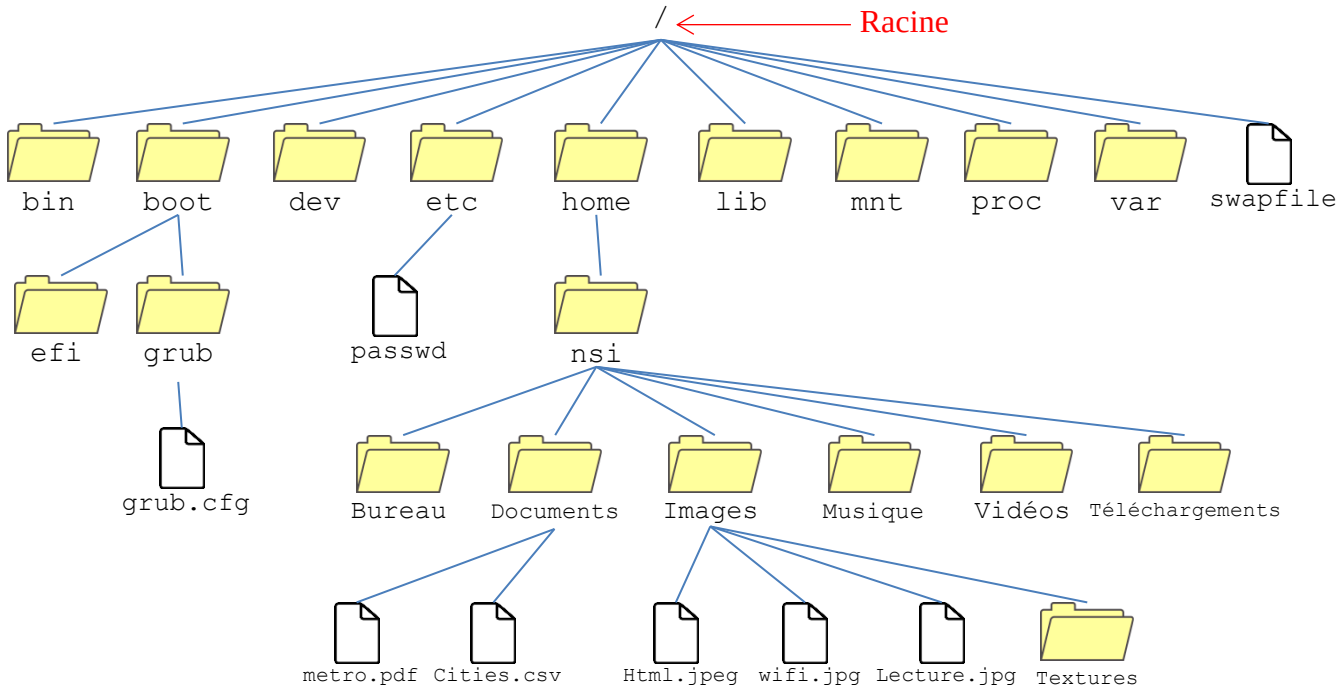
a. Arborescence

Les systèmes de fichiers modernes contiennent facilement plusieurs centaines de milliers de fichiers rien que pour un ordinateur personnel de base. Il est dès lors important d'avoir une organisation qui permette de s'y retrouver. Le système de fichier est donc organisé sous forme de nœuds qui peuvent être des fichiers ou des répertoires[2] et les répertoires peuvent eux-mêmes contenir des nœuds qui peuvent eux-mêmes être des sous-répertoires, etc…

[1] Sur des systèmes autre que les PC, ce logiciel est généralement appelé le *firmware*.
[2] Ou encore des liens ou des points de montage, mais cela sort du cadre de ce cours.

On obtient ainsi une organisation appelée « arborescence » en référence à la ressemblance avec un arbre dont les branches se subdivisent elles-mêmes en branches, etc …

On représente souvent l'arborescence d'un système de fichier à l'inverse des arbres en mettant la racine en haut et en descendant. On donne en exemple le schéma (très incomplet) suivant :



Sous linux, la racine du système de fichier se note « `/` »[3].

b. Chemins relatifs et absolus

Pour retrouver un fichier ou un dossier dans l'arborescence, on doit indiquer son **chemin**. Il existe deux façons de le faire :

- Le chemin absolu

  Celui-ci commence toujours de la racine et on liste tous les répertoires qu'il faut traverser pour arriver jusqu'au nœud voulu en les séparant par le signe « / ».

  Exemple : Pour accéder au fichier `grub.cfg` : `/boot/grub/grub.cfg`

  Pour accéder au fichier `wifi.jpg` : `/home/nsi/Images/wifi.jpg`

  On remarque qu'un chemin absolu commence toujours par le caractère « `/` ».

- Le chemin relatif

  Celui-ci commence non pas de la racine, mais d'un répertoire quelconque. On procède comme pour le chemin absolu, mais on retire la partie qui permet d'accéder au répertoire de départ.

  Exemple : Pour accéder au fichier `Lecture.jpg` depuis le répertoire `nsi` : `Images/wifi.jpg`

  Pour accéder au fichier `metro.pdf` depuis le répertoire `home` :

  `nsi/Documents/metro.pdf`

  On remarque qu'un chemin relatif ne commence jamais par le caractère « `/` ».

c. Répertoires spéciaux

Mais comment accéder au fichier `passwd` avec un chemin relatif si on est dans le répertoire `/boot/efi` ? Il faut pour cela utiliser les répertoires spéciaux « `.` » qui désigne le répertoire *courant* et « `..` » qui désigne le répertoire *parent* (c'est-à-dire juste au-dessus dans l'arborescence).

[3] Sous Windows, c'est une lettre de l'alphabet latin suivi du signe « `:\` ». Exemple « `c:\` »

Si on est dans le répertoire du fichier `Lecture.jpg`, alors « `.` » est le répertoire `Images` et « `..` » est le répertoire `nsi`.
Ainsi pour accéder au fichier `passwd` à partir du répertoire `/boot/efi`, le chemin relatif est `../../etc/passwd`.
De même pour accéder à `Html.jpeg` à partir du répertoire `Documents`, le chemin est `../Images/Html.jpeg`.

Il existe un dernier répertoire spécial : le répertoire personnel. Linux étant un OS multiutilisateurs, chaque utilisateur possède son propre répertoire (généralement situé dans le répertoire `/home`) qui correspond à son nom d'utilisateur. Ce répertoire est noté « `~` » ("tilde") dans le shell. Ainsi le chemin `~/Musique` correspond au répertoire `/home/nsi/Musique` pour l'utilisateur « `nsi` ». Mais pour l'utilisateur « `camille` », il correspondra à `/home/camille/Musique`.

Répertoire spécial	Signification
`/`	Racine du système de fichier
`.`	Répertoire courant (affichable avec la commande `pwd`). Il est également indiqué dans le prompt de l'invite de commande.
`..`	Répertoire parent du répertoire actuel. On peut l'utiliser plusieurs fois. Ainsi `../..` est le répertoire parent du répertoire parent.
`~`	Répertoire personnel de l'utilisateur. Il est généralement situé à `/home/`*`nom_utilisateur`*

Question 1 : Manipulation de chemins

1) Quel est le chemin absolu pour accéder aux fichiers situés dans le répertoire `Musique` de l'utilisateur `nsi` ? Vérifier votre réponse en listant le contenu de ce répertoire à l'aide de la commande `ls`.
2) Quelle commande permet de changer le répertoire courant ? Changer de répertoire pour aller dans le répertoire `/var/log`. Vérifier que le répertoire courant a bien changé avec la commande `pwd` et en notant que le prompt du shell a également changé.
3) Quel chemin **relatif** permet d'accéder aux fichiers du répertoire `Documents` de l'utilisateur `nsi` ? Vérifier votre réponse en listant le contenu de ce répertoire à l'aide de la commande `ls`.

Question 2 : Affichage du contenu

1) Quelle commande permet d'afficher le contenu d'un fichier texte ? Afficher dans la console le contenu du fichier `syslog` situé dans le répertoire `/var/log`.

Ce fichier est très long et on souhaite consulter uniquement les informations liée à l'ajout de l'extension de virtualbox.

2) Quelle commande permet de filtrer le contenu d'un texte ? Utiliser cette commande à l'aide d'un pipe (voir aide) pour n'afficher que les lignes de `syslog` qui contiennent le texte « `vboxadd` ».

On veut maintenant sauvegarder le résultat de la commande précédente dans un fichier « `vboxadd.log` » qu'on stockera dans le répertoire `/home/nsi/Documents`.

3) Ecrire la commande correspondante et tester sa bonne exécution en affichant le contenu du fichier « `vboxadd.log` ».

Aide :

- Une des caractéristiques du shell est que l'on peut rediriger la sortie d'une commande vers une autre commande. On appelle cela un « pipe » (tuyau en anglais) et c'est le caractère « `|` » (barre verticale) qui permet de rediriger.
  Ainsi `cmd1 | cmd2` permet de rediriger le résultat de cmd1 vers cmd2 (la sortie de cmd1 sera l'entrée de cmd2).

- On peut rediriger la sortie d'une commande vers un fichier plutôt que l'écran en utilisant l'opérateur « > » (signe « supérieur »). Par exemple `ls /boot > rep_boot.txt` va créer un fichier texte `rep_boot.txt` contenant le listing du répertoire `/boot`.

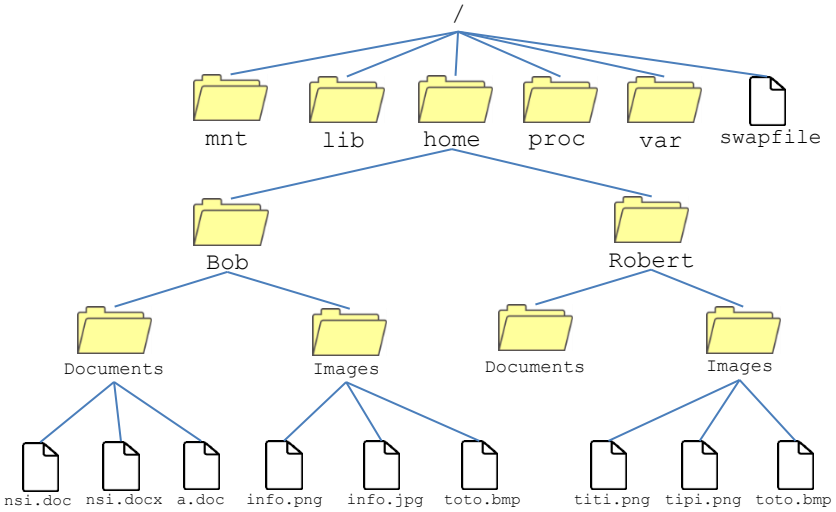
d. Caractères « jokers »

Lorsqu'on désigne un chemin on peut utiliser des « jokers » ce sont des caractères spéciaux qui ne doivent jamais être utilisés dans des noms de fichiers ou de dossiers et qui peuvent remplacer un ou plusieurs caractères :
« `?` » représente un (et un seul) caractère quelconque
« `*` » représente n'importe quel nombre (y compris 0) de caractères quelconques

Cela permet de désigner facilement un ou plusieurs fichiers ou répertoires dont le nom correspond à certains critères.

Exemples :
`*.jpg` désigne tous les fichiers ou répertoire finissant par « .jpg » (donc toutes les images jpeg)
`*super*` désigne tous les fichiers ou répertoire contenant « super » dans le nom.
`Résultats202?.csv` peut désigner les fichiers Résultats2020.csv, Résultats2021.csv, Résultats2022.csv, ... mais aussi Résultats202f.csv mais **pas** Résultats2021_avril.csv ni Résultat202.csv.

Prenons l'arborescence ci-contre :



Question 3 : Jokers
1) A l'intérieur de leur répertoire, comment désigner l'ensemble des 2 fichiers `nsi.doc` et `nsi.docx` ?
2) Dans le répertoire `/home/Robert/Images`, que désigne `ti?i.*` ?
3) Quels sont les fichiers désignés par `/home/?ob*/Images/*.png` ?

Question 4 : Jokers – mise en pratique
Quelle commande permet de lister tous les fichiers situés dans des sous-répertoires directs de `/home/nsi` et qui contiennent « NSI » (en majuscule) dans leur nom ? Exécutez la commande et noter le résultat (on doit obtenir 4 fichiers).

3) Droits d'accès aux fichiers et répertoires

Essayons de supprimer un fichier de configuration du système d'exploitation dans la console bash :

```
nsi@nsi-VirtualBox:~$ rm /etc/hosts
rm : supprimer '/etc/hosts' qui est protégé en écriture et est du type « fichier » ? o
rm: impossible de supprimer '/etc/hosts': Permission non accordée
```

On voit que le système refuse car la permission n'est pas accordée.

a. Droits des fichiers et des dossiers

Dans un système de fichier `ext` (le système de fichier standard de linux), chaque fichier ou répertoire possède des propriétés qui définissent les droits d'accès à ce fichier/répertoire.

Les fichiers et les répertoires possèdent 3 types de **droits** :

- les droits en lecture (symbolisés par la lettre r) : est-il possible de lire le contenu de ce fichier ?
- les droits en écriture (symbolisés par la lettre w) : est-il possible de modifier le contenu de ce fichier ?
- les droits en exécution (symbolisés par la lettre x) : est-il possible d'exécuter le contenu de ce fichier (quand le fichier est du code exécutable) ?

Il existe 3 types d'**utilisateurs** pour un fichier ou un répertoire :

- le propriétaire du fichier (par défaut c'est la personne qui a créé le fichier), il est symbolisé par la lettre "u".
- un fichier est associé à un groupe, tous les utilisateurs appartenant à ce groupe possèdent des droits particuliers sur ce fichier. Le groupe est symbolisé par la lettre "g".
- tous les autres utilisateurs (ceux qui ne sont pas le propriétaire du fichier et qui n'appartiennent pas au groupe associé au fichier). Ces utilisateurs sont symbolisés la lettre "o".

Regardons les propriétés du fichier « hosts » que nous n'avons pas eu le droit de supprimer. Pour cela nous allons utiliser la commande « `ls` » avec l'option « `-l` » afin d'avoir des informations supplémentaires.

```
nsi@nsi-VirtualBox:/etc$ ls -l hosts
-rw-r--r-- 1 root root 229 avril 22 22:50 hosts
```

Lisons ligne de gauche à droite :

- ⇨ le premier symbole "`-`" signifie que l'on a affaire à un fichier, dans le cas d'un répertoire, nous aurions un "`d`" à la place.
- ⇨ les 3 symboles suivants "`rw-`"donnent les droits du propriétaire du fichier : lecture autorisée (`r`), écriture autorisée (`w`), exécution interdite (`-` à la place de `x`).
- ⇨ les 3 symboles suivants "`r--`"donnent les droits du groupe auquel le fichier est associé : lecture autorisée (`r`), écriture interdite (`-` à la place de `w`), exécution interdite (`-` à la place de `x`).
- ⇨ les 3 symboles suivants "`r--`"donnent les droits des autres utilisateurs : lecture autorisée (`r`), écriture interdite (`-` à la place de `w`), exécution interdite (`-` à la place de `x`).
- ⇨ le caractère suivant "`1`" donne le nombre de liens (nous n'étudierons pas cette notion ici).
- ⇨ le premier "`root`" représente le nom du propriétaire du fichier.
- ⇨ le second "`root`" représente le nom du groupe lié au fichier.
- ⇨ le nombre "`229`" représente la taille du fichier en octet (ici notre fichier fait 229 octets).
- ⇨ "`avril 22 22:50`" donne la date et l'heure de la dernière modification du fichier.
- ⇨ "`hosts`" est le nom du fichier

Un répertoire est traité différemment. L'accès en lecture donne le droit de consulter la liste de ses entrées, l'accès en écriture celui d'y créer ou supprimer des fichiers et l'accès en exécution de le traverser (et notamment de s'y rendre avec la commande `cd`). Pouvoir traverser un répertoire sans le lire donne le droit d'accéder à celles de ses entrées dont on connaît le nom, mais pas de les trouver si on ignore leur existence ou leur nom exact.

Question 5 : Droits sur le répertoire nsi

Utiliser la commande `ls -l` sur le répertoire `/home/nsi` (ou `~`) et déterminer si les autres utilisateurs ont le droit de parcourir les fichiers de l'utilisateur nsi, de les lire, les effacer où créer de nouveaux fichiers dans les sous-répertoires.

b. Modification des droits

Pour modifier les droits d'un fichier ou d'un répertoire on peut utiliser la commande `chmod`.
Il y a deux façons de l'utiliser, mais nous nous contenterons de voir la plus simple. Pour chaque catégorie d'utilisateurs (`u`/`g`/`o`), on peut définir les droits (`=`), en ajouter (`+`), ou en retrancher (`-`). Ainsi, la formule

`u=rwx,g+rw,o-r` donne au propriétaire les droits de lecture, d'écriture et d'exécution ; ajoute au groupe propriétaire les droits de lecture et d'écriture ; et supprime le droit de lecture aux autres utilisateurs. Les droits non concernés par les opérations d'ajout ou de retranchement restent inchangés. La lettre `a`, pour *all*, recouvre les trois catégories d'utilisateurs, de sorte que `a=rx` donne aux trois catégories les mêmes droits (lecture et exécution, mais pas écriture).

Exemples :
`chmod o=rx consignes.txt` permet aux autres utilisateurs de lire le fichier `consignes.txt` mais pas de modifier son contenu.
`chmod g-x script.sh` retire aux utilisateurs du groupe le droit d'exécuter le fichier `script.sh`.

Question 6 : Modification des droits

1) Dans le répertoire de l'utilisateur nsi (`~`), créer un sous-répertoire `test` avec la commande `mkdir`. Dans le répertoire `test` créer un fichier `vide.txt` avec la commande `touch`. Regarder les droits sur le répertoire et le fichier crées avec `ls -l`.
2) Quelle commande faut-il taper pour faire en sorte que d'autres utilisateurs puissent créer des fichiers dans ce répertoire ? Modifier les droits en conséquence.

Question 7 : Création d'un nouvel utilisateur et test des droits

La commande pour créer un nouvel utilisateur est `adduser` *`nomutilisateur`*.

1) Essayer de créer un nouvel utilisateur nommé « `toto` ». Quel message obtenez-vous ?

On voit que les droits ne sont pas que sur les fichiers et répertoires mais qu'ils couvrent également d'autres domaines. Ici seul le super-utilisateur (`root`) a la possibilité d'exécuter cette commande.

2) Quelle commande que vous avez vue dans le jeu Terminus permet d'exécuter une autre commande en mode super-utilisateur ? Utiliser la commande pour appeler `adduser` et créer l'utilisateur `toto` (on lui donnera le mot de passe « `toto` »). Le mot de passe du compte `nsi` est « `Joliot-Curie` » (sans les guillemets et en respectant la casse).

Pour se connecter à une session shell sous l'utilisateur `toto`, on peut utiliser la commande

```
sudo -u toto -i
```

Le prompt du shell devrait refléter le changement de même que le répertoire personnel (`~`) doit maintenant être `/home/toto`. Pour sortir de cette session, on fait tout simplement la commande `exit`.

3) Utiliser la commande précédente pour passer sous l'utilisateur `toto` et vérifier que vous pouvez naviguer dans les répertoires de l'utilisateur `nsi` mais pas y créer de fichier à part dans le répertoire `test` crée précédemment. Attention : il y a une petite difficulté à solutionner sur le répertoire `/home/nsi` avant que cela fonctionne comme prévu.

Références :

Histoire des systèmes d'exploitation : https://www.youtube.com/watch?v=4OhUDAtmAUo
Histoire de linux : https://www.youtube.com/watch?v=IquNF_DXcF8
Un système d'exploitation c'est quoi ? : https://www.youtube.com/watch?v=YScMI8lsy9s
Description technique du processus de démarrage : https://www.mossywell.com/home/technical-stuff/boot-sequence