Exercice 1 – commandes shell de base :

Je me trouve dans le répertoire « `/home/nsi/` » et j'exécute les commandes suivantes :

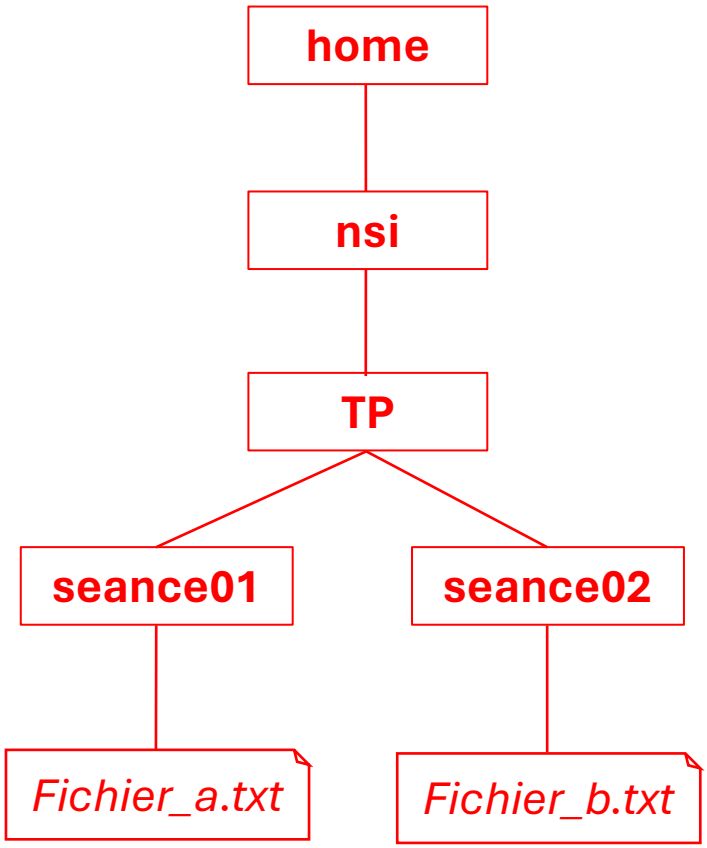
```
1. mkdir TP
2. cd TP
3. mkdir seance01
4. mkdir seance02
5. cd seance01
6. touch Fichier_a.txt
7. cd ..
8. cd seance02
9. touch Fichier_b.txt
```

➔ *Dessiner l'arborescence créée par ces commandes.*

Correction :

L'idée est de faire ce que je vous ai montré au tableau vendredi : partir de la base « `/home/nsi/` », la représenter, puis compléter le graphique de l'arborescence étape par étape en vérifiant à chaque fois « où » (donc dans quel répertoire) vous vous trouvez. Donc en l'occurrence à la ligne 1 on est dans `/home/nsi` ; aux lignes 2 à 4 on est dans `/home/nsi/TP` ; aux lignes 5 et 6 on est dans `/home/nsi/TP/seance01` ; à la 7 on est « remontés d'un cran » avec le « .. » donc on se trouve dans `/home/nsi/TP` ; et enfin à la 8 et 9 on est dans `/home/nsi/TP/seance02`.

L'arborescence qui résulte de tout ça (et donc la réponse à la question) est :



<u>À la suite de ces commandes</u> j'exécute les suivantes :

```
cd ../..
pwd
```

➔ *Quel sera le retour de cette commande (qu'est-ce qui va s'afficher dans la console) ?*

<u>Correction :</u>

Ce n'est pas pour rien que l'énoncé met en gras et souligné le « à la suite de ces commandes » ; le point de départ est donc le répertoire où on se trouvait à la fin de la question précédente, soit `/home/nsi/TP/seance02`. On monte alors de deux crans (deux fois « .. ») et on arrive donc dans `/home/nsi/`. La commande affiche le nom du répertoire où l'on se trouve. Son retour sera donc :

`/home/nsi/`

Quelqu'un a modifié le fichier `Fichier_a.txt` précédent et vous voulez en lire le contenu.

➔ *En une, deux, ou trois commandes successives **<u>à la suite des précédentes</u>** faites s'afficher dans la console le contenu du fichier `Fichier_a.txt`.*

<u>Correction :</u>

Même chose, les commandes sont « à la suite des précédentes ». Si l'on se réfère à l'arborescence, il s'agit d'afficher le contenu d'un fichier (commande « `cat` » comme on l'a vu en TP) qui est deux étages en-dessous d'où l'on se trouve.
En une commande :

```
cat TP/seance01/Fichier_a.txt
```

En deux commandes :

```
cd TP/seance01
cat Fichier_a.txt
```

En trois commandes :

```
cd TP
cd seance01
cat Fichier_a.txt
```

<u>Exercice 2 – commandes shell toujours :</u>

➔ Quelle commande taper pour simplement afficher la liste du contenu d'un répertoire ?

<u>Correction :</u>

```
ls
```

➔ Comment modifier cette commande pour qu'elle affiche aussi les droits des fichiers concernés ?

<u>Correction :</u>

```
ls -al
```

➔ Comment s'afficheront les droits du fichier « fichier.txt » si le propriétaire peut le lire et écrire dedans, et que tous les autres droits sont désactivés ?

Correction :

```
rw-------
```

➔ Quelle commande taper pour que les droits du fichier « fichier.txt » soient ceux-là (le propriétaire peut le lire et écrire dedans, et que tous les autres droits sont désactivés) ?

Correction :

```
chmod 600 fichier.txt
```

Si vous ne voyez pas comment je suis arrivé à ce « 600 » référez-vous à notre TP sur le shell où c'est expliqué.

Exercice 3 – questions de cours sur les réseaux :

➔ Quelles sont, du haut vers le bas, les quatre couches du modèle TCP/IP ?

Correction :

Application / Transport / Internet / Réseau

➔ Que signifie encapsuler des données dans le contexte d'un passage d'une couche à l'autre ?

Correction :

Cela signifie ajouter un en-tête contenant des données relatives à la couche dans laquelle nous nous trouvons.

➔ Que contient la couche la plus haute ?

Correction :

Les données, tout simplement.

➔ Que contient l'en-tête de la couche la plus basse quand on fait un transfert sur internet avec le protocole ethernet ? Y a-t-il un lien entre ce contenu et la notion de table ARP ?

Correction :

Cet en-tête contient les adresses MAC source et destination. Ces adresses sont obtenues par lecture de tables ARP qui fournissent la correspondance entre adresses IP et adresses MAC.

➔ Que signifie « passerelle par défaut » ?

Correction :

C'est le point de sortie du réseau dans lequel nous nous trouvons. Elle est utilisée lorsqu'un paquet est destiné à une machine qui ne se trouve pas dans le même réseau que nous (typiquement lorsque l'on envoie une requête http vers un serveur web par internet). L'analogie que l'on peut utiliser pour comprendre ce concept est celui du pont (la « passerelle ») qui relie notre île (le réseau) au reste du monde (internet).

Exercice 4 – adresses IP :

➔ Convertissez l'adresse IP suivante de la notation binaire à la notation décimale : `11000000.10101000.00000001.00000001` (on rappelle que $2^7 = 128$).

Correction :

On rappelle que chaque bit de chaque octet correspond à une puissance de deux, de 2^0 à droite jusqu' à 2^7 à gauche. Donc par exemple, pour le deuxième octet ici on aura :
$(10101000)_2 = 1x2^7 + 0x2^6 + 1x2^5 + 0x2^4 + 1x2^3 + 0x2^2 + 0x2^1 + 0x2^0 = 128 + 32 + 8 = 168$
En appliquant cette même logique aux trois autres octets on obtient l'adresse : 192.168.1.1

➔ Déterminez l'adresse réseau de l'hôte avec l'adresse IP `192.168.10.15` et le masque de sous-réseau `255.255.255.255`.

Correction :

Ce masque n'a pas de sens puisqu'il n'est constitué que de 255 en décimal, donc que de 1 en binaire. L'adresse réseau sera donc dans ce cas l'adresse elle-même : 192.168.10.15.

➔ Déterminez l'adresse réseau de l'hôte avec l'adresse IP `192.168.10.15` et le masque de sous-réseau `255.255.255.252`.

Correction :

Les trois premiers octets ne sont constitués que de 1, donc on sait déjà que les trois premières valeurs de l'adresse réseau seront identiques à ceux de l'adresse IP : 192.168.10.

Pour le quatrième, il s'agit de convertir sa valeur décimale en binaire. La méthode la plus simple consiste à prendre les bits les uns après les autres et voir ce que cela donne :

- 11111111 correspond à 255 ;
- Si je passe le dernier 1 à 0, j'enlève 2^0, donc 1, donc 11111110 correspond à 254 ;
- Si je passe le 1 suivant à 0, j'enlève 2^1, donc 2, donc 11111100 correspond à 252 ;

Le dernier octet de l'adresse réseau recherchée est le même que celui de l'adresse IP à l'exception de ses deux derniers bits.

Pour appliquer mon « ET » entre ce masque et l'adresse IP il faut que je convertisse le dernier octet (valeur : 15) en binaire. Ça, vous savez le faire (on l'a appris plus tôt dans l'année) mais en l'occurrence c'est très simple puisque $15 = 16 - 1 = 2^4 - 1$. Ecrit en binaire cela veut dire que : $15 = (10000 - 1)_2 = (1111)_2$.

Donc le dernier octet de l'adresse IP est : 00001111. Il ne nous reste qu'à appliquer le « ET » logique avec l'octet correspondant à 252 que l'on a calculé plus haut :

```
        0 0 0 0 1 1 1 1
ET      1 1 1 1 1 1 0 0
=       0 0 0 0 1 1 0 0
```

On convertit ça en décimal ($2^3 + 2^2 = 8 + 4 = 12$) et on obtient donc l'adresse réseau complète : 192.168.10.12.

➔ Déterminez l'adresse réseau de l'hôte avec l'adresse IP `192.168.10.15` et le masque de sous-réseau `255.255.255.248`.

Correction :

EXACTEMENT le même raisonnement qu'à la question précédente sauf que le dernier octet vaut maintenant 248. En résumé rapide les étapes sont :

- Dernier octet du masque en binaire : 11111000
- Résultat de 11111000 ET 00001111 : 00001000
- Conversion en décimal : $2^3 = 8$.
- Adresse du réseau : 192.168.10.8.

➔ Déterminez l'adresse réseau de l'hôte avec l'adresse IP `192.168.10.15` et le masque de sous-réseau `255.255.255.240`.

Correction :

Encore la même chose :

- Dernier octet du masque en binaire : 11110000
- Résultat de 11110000 ET 00001111 : 00000000
- Conversion en décimal : 0.
- Adresse du réseau : 192.168.10.0.

Exercice 5 – divers réseau :

➔ Pour le réseau `192.168.5.0/24`, donnez l'adresse du premier hôte et du dernier hôte possible.

Correction :

Cette notation signifie que les 24 premiers bits (donc les trois premiers octets) constituent le masque et donc l'adresse réseau pour cette adresse IP. Les adresses IP disponibles sont donc 192.168.5.0 jusqu'à 192.168.5.255 (*NB : ce n'est pas strictement vrai, la première et la dernière adresse sont en fait réservées – mais cette réponse à la question est tout à fait acceptable néanmoins*).

➔ Le masque de sous réseau `255.255.4.0` a-t-il un sens ? Pourquoi ?

Correction :

Non il n'a pas de sens, parce que converti en binaire il donne : 11111111.11111111.00000100.00000000. Or on sait qu'un masque est une série de bits de gauche à droite positionnés à 1. Ce 1 au milieu du troisième octet ne peut fonctionner.

➔ Soit le masque `255.255.254.0`. Parmi les adresses IP suivantes, lesquelles appartiennent au même réseau ?
`192.168.12.1 ; 192.168.10.1 ; 192.168.11.50 ; 192.168.14.2 ; 192.168.10.255`.

Correction :

Je ne refais pas tous les calculs en détail – vous devriez en être capables tous seuls (et demandez à ChatGPT de vous aider au besoin !!).

Masque de sous réseau : 11111111.11111111.11111110.00000000

On ne s'intéresse donc qu'au troisième octet des adresses IP (les trois autres dans le masque étant soit intégralement à 0 soit intégralement à 1).

Masque :	11111110
IP1 :	12
IP1 :	00001100
Résultat du ET :	00001100 = 12

Adresse réseau de l'IP1 : 192.168.12.0

Masque : 11111110
IP2 : 10
IP2 : 00001010
Résultat du ET : 00001010 = 10
Adresse réseau de l'IP2 : 192.168.10.0

Masque : 11111110
IP3 : 11
IP3 : 00001011
Résultat du ET : 00001010 = 10
Adresse réseau de l'IP1 : 192.168.10.0

Masque : 11111110
IP4 : 14
IP4 : 00001110
Résultat du ET : 00001110 = 14
Adresse réseau de l'IP1 : 192.168.14.0

Masque : 11111110
IP5 : 10
IP5 : 00001010
Résultat du ET : 00001010 = 10
Adresse réseau de l'IP1 : 192.168.10.0

Les adresses appartenant au même réseau, dont l'adresse est 192.168.10.0, sont donc : 192.168.10.1, 192.168.11.50, et 192.168.10.255.