NOLLE

Damien

L3 – Informatique

SR – Devoir 2 :

Note :	***Observation :***
/20	

Exercice 1)

Q1 :

Etape 0 : Accès au compte « root » avec la commande « su » et le mot de passe « EadSr2017 » :

```
user@cours:~$ su
Mot de passe :
root@cours:/home/user#
```

Etape 1 : Création des répertoires « licence3 », « reinscription », « enseignement », « accesSSH » (correspondant aux groupes), « alice », « charlie », « bob » et « eve » (correspondant au utilisateur avec la commande « mkdir » :

```
root@cours:/home# mkdir /home/licence3 /home/reinscription /home/enseignement /home/accesSSH /home/a
lice /home/charlie /home/bob /home/eve
root@cours:/home# ls
accesSSH     challenge01  challenge05  challenge09  challenge13  enseignement  user
alice        challenge02  challenge06  challenge10  challenge14  eve
bob          challenge03  challenge07  challenge11  challenge15  licence3
challenge00  challenge04  challenge08  challenge12  charlie      reinscription
```

Pour l'instant, tout les répertoires précédemment créés ont pour propriétaire « root » et pour groupe, le groupe principal du super-utilisateur nommé « root ».

Etape 2 : Création des différents groupes (licence3, reinscription, enseignement et accesSSH) avec la commande « groupadd » :

```
root@cours:~# groupadd licence3
root@cours:~# groupadd reinscription
root@cours:~# groupadd enseignement
root@cours:~# groupadd accesSSH
root@cours:~# cat /etc/group | grep -E '(licence3|reinscription|enseignement|accesSSH)'
licence3:x:1017:
reinscription:x:1018:
enseignement:x:1019:
accesSSH:x:1020:
```

Etape 3 : Création des différents utilisateurs (alice, charlie, bob et eve) avec la commande « useradd », en sachant que tous les utilisateurs auront comme groupe principal, le groupe accesSSH et que alice appartient au groupe licence3 et reinscription, charlie au groupe licence3, bob au groupe reinscription et eve au groupe enseignement :

```
root@cours:~# useradd -g accesSSH -G licence3,reinscription -d /home/alice -c Etudiante alice
root@cours:~# useradd -g accesSSH -G licence3 -d /home/charlie -c Etudiant charlie
root@cours:~# useradd -g accesSSH -G reinscription -d /home/bob -c Etudiant bob
root@cours:~# useradd -g accesSSH -G enseignement -d /home/eve -c Enseignante eve
```

```
root@cours:~# cat /etc/passwd | grep -E '(alice|charlie|bob|eve)'
alice:x:1017:1020:Etudiante:/home/alice:/bin/sh
charlie:x:1018:1020:Etudiant:/home/charlie:/bin/sh
bob:x:1019:1020:Etudiant:/home/bob:/bin/sh
eve:x:1020:1020:Enseignante:/home/eve:/bin/sh
root@cours:~# cat cat /etc/group | grep -E '(licence3|reinscription|enseignement|accesSSH)'
licence3:x:1017:alice,charlie
reinscription:x:1018:alice,bob
enseignement:x:1019:eve
accesSSH:x:1020:
```

Etape 4 : Changement des groupes sur les répertoires « licence3 », « reinscription », « enseignement » et « accesSSH » avec les groupes correspondants via la commande « chown » :

```
root@cours:~# chown :licence3 /home/licence3
root@cours:~# chown :reinscription /home/reinscription
root@cours:~# chown :enseignement /home/enseignement
root@cours:~# chown :accesSSH /home/accesSSH
```

```
root@cours:~# ls -l /home | grep -E '(enseignement|licence3|reinscription|accesSSH)'
drwxr-xr-x 2 root        accesSSH      4096 janv.  7 02:01 accesSSH
drwxr-xr-x 2 root        enseignement  4096 janv.  7 02:01 enseignement
drwxr-xr-x 2 root        licence3      4096 janv.  7 02:01 licence3
drwxr-xr-x 2 root        reinscription 4096 janv.  7 02:01 reinscription
```

Seulement le groupe a été changé, le propriétaire restera donc « root ».

Etape 5 : Changement des groupes et des utilisateurs propriétaire pour chaque répertoire personnel en fonction de l'utilisateur correspondant avec la commande « chown » :

```
root@cours:~# chown eve:accesSSH /home/eve
root@cours:~# chown charlie:accesSSH /home/charlie
root@cours:~# chown bob:accesSSH /home/bob
root@cours:~# chown alice:accesSSH /home/alice
```

```
root@cours:~# ls -l /home | grep -E '(eve|charlie|bob|alice)'
drwxr-xr-x 2 alice       accesSSH      4096 janv.  7 02:01 alice
drwxr-xr-x 2 bob         accesSSH      4096 janv.  7 02:01 bob
drwxr-xr-x 2 charlie     accesSSH      4096 janv.  7 02:01 charlie
drwxr-xr-x 2 eve         accesSSH      4096 janv.  7 02:01 eve
```

On attribue le groupe principal de tous les utilisateurs (à savoir « accesSSH ») de manière à attribuer des droits pour tous les utilisateurs du serveur de fichier, afin que seul l'utilisateur propriétaire n'y ai accès. En réalité, le groupe n'a pas d'importance ici car, quel que soit le groupe d'appartenance, seul le propriétaire y aura accès. Donc les droits pour le groupe seront les mêmes que pour les autres.

Etape 6 : Changement des droits des répertoires personnels pour que seul l'utilisateur propriétaire ai un accès :

Pour cela, il faut que les propriétaires puissent afficher (r), modifier (w) et explorer le contenu du répertoire.

Il faut alors attribuer les droits :

- Le propriétaire (u) : « rwx » ou $2^2 * 1 + 2^1 * 1 + 2^0 * 1 = 4 + 2 + 1 = 7$ en octal.
- Le groupe (g) : « » ou $2^2 * 0 + 2^1 * 0 + 2^0 * 0 = 0 + 0 + 0 = 0$ en octal.
- Les autres (o) : « » ou $2^2 * 0 + 2^1 * 0 + 2^0 * 0 = 0 + 0 + 0 = 0$ en octal.

On attribut ensuite les droits avec la commande « chmod », soit en écriture octal, soit en écriture rwx :

```
root@cours:~# chmod u=rwx,g=,o=g /home/alice
root@cours:~# chmod u=rwx,g=,o=g /home/bob
root@cours:~# chmod u=rwx,g=,o=g /home/charlie
root@cours:~# chmod u=rwx,g=,o=g /home/eve
```

```
root@cours:~# ls -l /home | grep -E '(eve|charlie|bob|alice)'
drwx------ 2 alice       accesSSH      4096 janv.  7 02:01 alice
drwx------ 2 bob         accesSSH      4096 janv.  7 02:01 bob
drwx------ 2 charlie     accesSSH      4096 janv.  7 02:01 charlie
drwx------ 2 eve         accesSSH      4096 janv.  7 02:01 eve
```

Etape 7 : Changement des droits des répertoires de chaque groupe pour que seul le groupe propriétaire et l'utilisateur propriétaire (à savoir « root ») ai un accès :

Pour cela, il faut que le propriétaire et les membres du groupe puissent afficher (r), modifier (w) et explorer le contenu du répertoire.

Il faut alors attribuer les droits :

- Le propriétaire (u) : « rwx » ou $2^2 * 1 + 2^1 * 1 + 2^0 * 1 = 4 + 2 + 1 = 7$ en octal.
- Le groupe (g) : « rwx » ou $2^2 * 1 + 2^1 * 1 + 2^0 * 1 = 4 + 2 + 1 = 7$ en octal.
- Les autres (o) : « » ou $2^2 * 0 + 2^1 * 0 + 2^0 * 0 = 0 + 0 + 0 = 0$ en octal.

On attribut ensuite les droits avec la commande « chmod », soit en écriture octal, soit en écriture rwx :

```
root@cours:~# chmod u=rwx,g=u,o= /home/accesSSH
root@cours:~# chmod u=rwx,g=u,o= /home/licence3/
root@cours:~# chmod u=rwx,g=u,o= /home/reinscription
root@cours:~# chmod u=rwx,g=u,o= /home/enseignement
```

```
root@cours:~# ls -l /home | grep -E '(enseignement|licence3|reinscription|accesSSH)' | grep -E 'root
'
drwxrwx--- 2 root        accesSSH      4096 janv.  7 02:01 accesSSH
drwxrwx--- 2 root        enseignement  4096 janv.  7 02:01 enseignement
drwxrwx--- 2 root        licence3      4096 janv.  7 02:01 licence3
drwxrwx--- 2 root        reinscription 4096 janv.  7 02:01 reinscription
```

Q2 :

Script permettant de demander à l'utilisateur un répertoire dans « home » vers lequel se déplacer, si rien n'est saisi, il reste dans son répertoire personnel :

```
 GNU nano 3.2                          /scripts/changed.sh

#!/bin/bash

echo -e "\nQuel répertoire voulez-vous accéder ?"
printf "> "

read repertoire

if [ ! -z "$repertoire" -a ! -d "/home/$repertoire" ]
then
        while [ ! -z "$repertoire" -a ! -d "/home/$repertoire" ]
        do
                echo "Erreur : Veuillez saisir le nom d'un réperoire valide !"
                printf "Nouvelle saisie > "

                read repertoire
        done
fi

if [ ! -z "$repertoire" ]
then
        cd /home/$repertoire
else
        cd $HOME
fi

                        [ Lecture de 24 lignes ]
^G Aide       ^O Écrire     ^W Chercher   ^K Couper     ^J Justifier  ^C Pos. cur.  M-U Annuler
^X Quitter    ^R Lire fich. ^\ Remplacer  ^U Coller     ^T Orthograp. ^_ Aller lig. M-E Refaire
```

Pour que ce script soit lancé à la connexion d'un utilisateur, il faut soit écrire le script dans le fichier « /etc/profile », soit permettre l'exécution du script via un fichier de commande depuis le fichier « /etc/profile ».

Prenons ce dernier cas, on va venir mettre le script bash dans un répertoire ayant les permissions pour que les autres utilisateurs puissent lire et explorer le contenu de celui-ci :

```
root@cours:~# mkdir /scripts
root@cours:~# ls -l / | grep scripts
drwxr-xr-x  2 root root  4096 janv.  7 20:18 scripts
```

On remarque que les bonnes permissions sont déjà accordées à la création du répertoire, on n'a donc pas besoin de les modifier.

On déplace le script dans le répertoire précédemment créé et on ajoute la permission d'exécution pour tout le monde :

```
root@cours:~# mv changed.sh /scripts
root@cours:~# chmod a+x /scripts/changed.sh
root@cours:~# ls -l /scripts
total 4
-rwxr-xr-x 1 root root 412 janv.  7 19:44 changed.sh
```

On vient modifier le fichier « /etc/profile » de manière à modifier la variable d'environnement PATH pour ajouter le répertoire où se trouve le script :

```
 GNU nano 3.2                                    /etc/profile

# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).

if [ "`id -u`" -eq 0 ]; then
  PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
else
  PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
fi

PATH="$PATH:/scripts"

export PATH
```

Puis dans ce même fichier, on vient ajouter la commande pour exécuter le script (à savoir le nom du fichier, soit : « changed.sh ») avec la commande « . » afin d'exécuter le script dans le processus shell courant pour permettre le déplacement avec la commande « cd » :

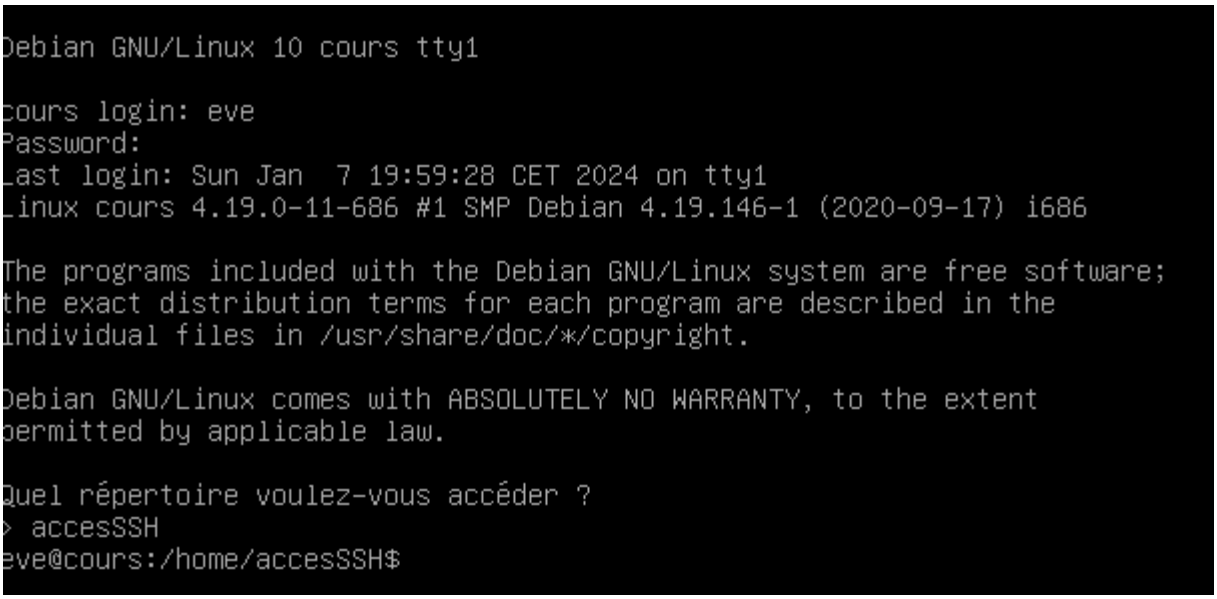
```
 GNU nano 3.2                                    /etc/profile

    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi

if [ -d /etc/profile.d ]; then
  for i in /etc/profile.d/*.sh; do
    if [ -r $i ]; then
      . $i
    fi
  done
  unset i
fi

. changed.sh
```

On redémarre la machine avec la commande « reboot » et on se connecte avec un utilisateur, par exemple, « eve » :



Q3 :

Les unmask permettent de changer les permissions par défaut appliquées lors de la création d'un nouveau fichier ou répertoire.

Pour déterminer les permissions appliquées, il faut réaliser les calculs suivants en fonction de si c'est :

- Un fichier :
  1) Complément à 1.
  2) On retire la permissions d'exécution à 1 en le mettant à 0.

- Un répertoire :
  1) Complément à 1.

Prenons les 3 différents umask que l'on peut positionner pour les utilisateurs :

- 044 :
  044 en octal donne : 000 100 100.

  1) Pour un fichier :
     Complément à 1 : 111 011 011.

     On retire les permission d'exécution : 110 010 010.

  2) Pour un répertoire :
     Complément à 1 : 111 011 011.

Lorsqu'un utilisateur créera un nouveau fichier, il pourra le lire et le modifier. Alors que son groupe et les autres utilisateurs pourront uniquement le modifier. Lorsqu'il créera un nouveau répertoire, il pourra afficher, modifier et explorer son contenu. Alors que son groupe et les autres utilisateurs pourront modifier et explorer son contenu.

- 066 :
  066 en octal donne : 000 110 110

  1) Pour un fichier :
     Complément à 1 : 111 001 001.

     On retire les permission d'exécution : 110 000 000.

  2) Pour un répertoire :
     Complément à 1 : 111 001 001.

  Lorsqu'un utilisateur créera un nouveau fichier, il pourra le lire et le modifier. Alors que son groupe et les autres utilisateurs n'auront aucune permission dessus et donc n'auront aucun accès. Lorsqu'il créera un nouveau répertoire, il pourra afficher, modifier et explorer son contenu. Alors que son groupe et les autres utilisateurs pourront uniquement explorer son contenu.

- 077 :
  077 en octal donne : 000 111 111

  1) Pour un fichier :
     Complément à 1 : 111 000 000.

     On retire les permission d'exécution : 110 000 000.

  2) Pour un répertoire :
     Complément à 1 : 111 000 000.

  Lorsqu'un utilisateur créera un nouveau fichier, il pourra le lire et le modifier. Alors que son groupe et les autres utilisateurs n'auront aucune permission dessus et donc n'auront aucun accès. Lorsqu'il créera un nouveau répertoire, il pourra afficher, modifier et explorer son contenu. Alors que son groupe et les autres utilisateurs n'auront aucune permission dessus et donc n'auront aucun accès.

Dans les trois cas, il s'agit de restreindre l'accès aux fichiers et aux répertoires par défaut pour le partage. Dans le premier cas (044), les fichiers pourront être modifié et le contenu des répertoires pourront être explorer et modifier par les autres utilisateurs et ceux du groupe. Dans le second cas

(066), ils ne pourront qu'explorer les répertoires mais ne pourront pas modifier le contenu ni l'afficher et n'auront aucuns accès aux fichiers. Dans le dernier cas (077), les fichiers et les répertoires ne seront accessibles que par l'utilisateur propriétaire.

Exercice 2)

Q1 :

Création du répertoire de travail, soit du répertoire « Devoir2NOLLE » et dans ce dernier, du répertoire « Exercice2NOLLE ». Le tout dans notre répertoire d'accueil avec la commande « mkdir » :

```
root@cours:~# mkdir ~/Devoir2NOLLE && mkdir ~/Devoir2NOLLE/Exercice2NOLLE
```

```
root@cours:~# ls -Rl
.:
total 4
drwxr-xr-x 3 root root 4096 janv.  7 21:52 Devoir2NOLLE

./Devoir2NOLLE:
total 4
drwxr-xr-x 2 root root 4096 janv.  7 21:52 Exercice2NOLLE

./Devoir2NOLLE/Exercice2NOLLE:
total 0
```

Q2 :

1) Pour afficher le PID et le PPID de chaque processus actif sur la machine, on peut utiliser la commande « ps -ef » :

```
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 20:29 ?        00:00:04 /sbin/init
root           2     0  0 20:29 ?        00:00:00 [kthreadd]
root           3     2  0 20:29 ?        00:00:00 [rcu_gp]
root           4     2  0 20:29 ?        00:00:00 [rcu_par_gp]
root           6     2  0 20:29 ?        00:00:00 [kworker/0:0H-kblockd]
root           8     2  0 20:29 ?        00:00:00 [mm_percpu_wq]
root           9     2  0 20:29 ?        00:00:00 [ksoftirqd/0]
root          10     2  0 20:29 ?        00:00:02 [rcu_sched]
root          11     2  0 20:29 ?        00:00:00 [rcu_bh]
```

2) Script « pereProc » permettant d'afficher pour chaque processus actif sur la machine, le processus père sous la forme « Le processus [PID] à pour père le processus [PPID] » :

```
  GNU nano 3.2                                  pereProc

#!/bin/bash

cpt=0

ps -ef | while read ligne
do
        if [ "$cpt" -gt 0 ]
        then
                echo "Le processus $(echo $ligne | cut -f2  -d" ") a pour père le processus $(echo
                $ligne | cut -f3  -d" ")"
        fi

        cpt=$((cpt+1))
done
```

Résultat :

```
Le processus 1 a pour père le processus 0
Le processus 2 a pour père le processus 0
Le processus 3 a pour père le processus 2
Le processus 4 a pour père le processus 2
Le processus 6 a pour père le processus 2
Le processus 8 a pour père le processus 2
Le processus 9 a pour père le processus 2
Le processus 10 a pour père le processus 2
Le processus 11 a pour père le processus 2
Le processus 12 a pour père le processus 2
Le processus 14 a pour père le processus 2
Le processus 15 a pour père le processus 2
```

Q3 : Script « statutProc » permettant d'afficher pour chaque processus actif sur la machine, son statut et à quoi il correspond :

```
 GNU nano 3.2                                  satutProc

#!/bin/bash

# /!\ ATTENTION /!\
# Ce script ne prend en compte que les status possibles sur les distributions Debian !
# Des changements seront donc à envisager en fonction de votre distribution Linux !

cpt=0

ps -aux | while read ligne
do
        if [ "$cpt" -gt 0 ]
        then
                case $(echo $ligne | cut -f8 -d" " | cut -c1) in
                "R") statut="Prêt à s'exécuter.";;
                "T") statut="Stoppé.";;
                "S") statut="Dormant depuis moins de 20 secondes.";;
                "I") statut="Dormant depuis plus de 20 secondes.";;
                "Z") statut="Processus zombie.";;
                "W") statut="Swappé.";;
                "0") statut="Processus orphelin.";;
                *) statut="Statut inconnu."
                esac

                echo "Le processus $(echo $ligne | cut -f2 -d" ") a pour statut : $(echo $ligne |
                cut -f8 -d" " | cut -c1) ($statut)"
        fi

        cpt=$((cpt+1))
done
```

Résultat :

```
Le processus 5011 a pour statut : I (Dormant depuis plus de 20 secondes.)
Le processus 7123 a pour statut : T (Stoppé.)
Le processus 7126 a pour statut : I (Dormant depuis plus de 20 secondes.)
Le processus 8069 a pour statut : I (Dormant depuis plus de 20 secondes.)
Le processus 9274 a pour statut : I (Dormant depuis plus de 20 secondes.)
Le processus 9276 a pour statut : I (Dormant depuis plus de 20 secondes.)
Le processus 9277 a pour statut : S (Dormant depuis moins de 20 secondes.)
Le processus 9278 a pour statut : R (Prêt à s'exécuter.)
Le processus 9279 a pour statut : S (Dormant depuis moins de 20 secondes.)
```

Q4 :

1) Script permettant d'afficher le PID du processus le plus prioritaire avec sa priorité :

```
 GNU nano 3.2                                  prioriteProc

#!/bin/bash

tab=($(ps -ef | sort -gk4 | head -n 2 | tail -n 1))

echo "Le processus le plus prioritaire a pour PID ${tab[1]} avec la priorité ${tab[3]}."
```

Résultat :

```
root@cours:~/Devoir2NOLLE/Exercice2NOLLE# bash prioriteProc
Le processus le plus prioritaire a pour PID 287 avec la priorité 0.
```

2) Pour afficher le PID du processus le plus prioritaire et sa priorité, on peut utiliser la commande « ps –efo "%p %n" | sort -gk2 | head -n 2 » :

```
root@cours:~/Devoir2NOLLE/Exercice2NOLLE# ps -efo "%p %n" | sort -gk2 | head -n 2
  PID  NI
10517   0
```

Exercice 3)

Q1 : On sait qu'il y aura 3 sous-réseaux : SR1 (la direction et le marketing), SR2 (l'unité de production) et SR3 (le bureau d'études). Chaque sous-réseaux contiendra un routeur.

- Pour SR1 : Il y aura 31 postes. Donc il faut 31 adresses IP pour ces postes + 1 adresse IP pour le routeur + 2 adresses IP réservés (correspondant à l'adresse réseau et de broadcast). Ce qui fait au total : 31 + 1 + 2 = 34 adresses IP qui seront utilisées.

- Pour SR2 : Il y aura 247 postes. Donc il faut 247 adresses IP + 1 adresse IP pour le routeur + 2 adresses IP réservés. Ce qui fait au total : 247 + 1 + 2 = 250 adresses IP qui seront utilisées.

- SR3 : Il y aura 450 postes. Donc il faut 450 adresses IP + 1 adresse IP pour le routeur + 2 adresses IP réservés. Ce qui fait au total : 450 + 1 + 2 = 453 adresses IP qui seront utilisées.

On sait que la taille d'un sous-réseau correspond au nombre d'adresses IP au total - 2 = nombre d'adresses IP disponibles.

Le nombre d'adresses IP total du sous-réseaux correspond à : $2^{\text{nombre de 0 du masque}}$.

Par exemple : Si dans le masque nous avons que deux 0, on aura alors $2^2 = 4$ adresses IP au total dans le sous-réseaux. Sois 4 – 2 = 2 adresses IP disponibles.

- Pour SR1 : $2^6 = 64 > 34$ adresses IP nécessaires mais $2^5 = 32 < 34$ adresses IP nécessaires. On retient donc 2^6 car sinon on n'aura pas assez d'adresses IP. Le sous-réseau aura donc une taille de 64 adresses IP au total pour 62 adresses IP disponibles pour les postes et le routeur.

- Pour SR2 : $2^8 = 256 > 250$ adresses IP nécessaires car $2^7 = 128 < 250$. Le sous-réseau aura donc une taille de 256 adresses IP au total pour 253 adresses IP disponibles pour les postes et le routeur.

- Pour SR3 : $2^9 = 512 > 453$ car $2^8 = 256 < 453$. Le sous-réseau aura donc une taille de 512 adresses IP au total pour 510 adresses IP disponibles pour les postes et le routeur.

Q2 : Le réseau 172.18.0.0 est de classe B car 128 < 172 < 191, son masque par défaut est donc 255.255.0.0.

Le réseau 10.0.0.0 est de classe A car 0 < 10 < 127, son masque par défaut est donc 255.0.0.0.

Le réseau 192.168.4.0 est de classe C car 192 = 192 < 223, son masque par défaut est donc 255.255.255.0.

Puisqu'il faut découper le réseau de base en 3 sous-réseaux, il faut donc déjà que ce réseau puisse accueillir les 64 + 256 + 512 = 832 adresses IP nécessaires pour se découpage.

Le premier réseau de base (Classe B) :

Le masque 255.255.0.0 donne : 11111111 11111111 00000000 00000000 en base 2. On sait que $2^{\text{nombre de 0 du masque}}$ correspond au nombre d'adresse IP total du réseau, donc : $2^{16} = 65536 > 832$ adresse IP nécessaires.

Le deuxième réseau de base (Classe A) :

Le masque 255.0.0.0 donne : 11111111 00000000 00000000 00000000 en base 2. On sait que $2^{\text{nombre de 0 du masque}}$ correspond au nombre d'adresse IP total du réseau, donc : $2^{24} = 16777216 > 832$ adresse IP nécessaires.

Le troisième réseau de base (Classe C) :

Le masque 255.255.255.0 donne : 11111111 11111111 11111111 00000000 en base 2. On sait que $2^{\text{nombre de 0 du masque}}$ correspond au nombre d'adresse IP total du réseau, donc : $2^8 = 256 < 832$ adresse IP nécessaires.

Le deuxième réseau de base (Classe A) à un nombre d'adresses IP total trop supérieur au nombre d'adresses nécessaires. A l'inverse, le troisième réseau de base (Classe C) à un nombre d'adresses IP total trop inférieur au nombre d'adresses nécessaires.

Le réseau retenu sera donc le premier réseau de base (Classe B), soit 172.18.0.0 /16.

Q3 :

Le masque du réseau est 255.255.0.0 soit 11111111 11111111 00000000 00000000 avec 172.18.0.0 comme adresse réseau.

- SR1 :

Puisque $2^6 = 64$ et que dans le masque de base on a : 16 bits à 0. Il faut donc ajouter 16 – 6 = 10 bits à 1 pour obtenir le masque de sous-réseau : 11111111 11111111 11111111 11000000, soit 255.255.255.($2^7 + 2^6 = 128 + 64 = 192$) en décimal pointé, soit /26 en notation CIDR (qui correspond au nombre de bits à 1 du masque).

En partant du début : 172.18.0.0 correspond à l'adresse IP de ce premier sous-réseau.

Alors 172.18.0.(0 + 64 = 64) correspond à l'adresse du prochain réseau. Donc 172.18.0.(64 – 1 = 63) correspond à l'adresse de diffusion (ou de broadcast).

La première adresse disponible est donc 172.18.0.(0 + 1 = 1).

La dernière adresse disponible est donc 172.18.0.(63 – 1 = 62).

- SR2 :

Puisque $2^8 = 256$ et que dans le masque de base on a : 16 bits à 0. Il faut donc ajouter 16 – 8 = 8 bits à 1 pour obtenir le masque de sous-réseau : 11111111 11111111 11111111 00000000, soit 255.255.255.0 en décimal pointé, soit /24 en notation CIDR.

On a déterminé précédemment le masque du prochain sous-réseau : 172.18.0.64 correspond à l'adresse IP de ce second sous-réseau.

Alors 172.18.(0 + 1 = 1).(64 + 256 = 320 sauf qu'il n'est pas possible d'avoir 320 comme valeur pour un octet, on fait donc 320 – 256 = 64 et on ajoute 1 à l'octet précédent, soit le troisième), ce qui nous donne : 172.18.1.64, correspond à l'adresse du prochain réseau. Donc 172.18.1.(64 – 1 = 63) correspond à l'adresse de diffusion (ou de broadcast).

La première adresse disponible est donc 172.18.0.(64 + 1 = 65).

La dernière adresse disponible est donc 172.18.1.(63 – 1 = 62).

- SR3 :

Puisque $2^9 = 512$ et que dans le masque de base on a : 16 bits à 0. Il faut donc ajouter 16 – 9 = 7 bits à 1 pour obtenir le masque de sous-réseau : 11111111 11111111 11111110 00000000, soit 255.255.($2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 = 128 + 64 + 32 + 16 + 8 + 4 + 2 \Leftrightarrow 255 – 1 = 254$).0 en décimal pointé, soit /23 en notation CIDR.

On a déterminé précédemment le masque du prochain sous-réseau : 172.18.1.64 correspond à l'adresse IP de ce troisième sous-réseau.

Alors 172.18.(1 + 2 = 3).(64 + 512 = 576 ce qui n'est pas possible comme valeur pour un octet, donc 576 – 256 = 320 ce qui n'est toujours pas possible, donc 320 – 256 = 64 et on ajoute 2 à l'octet précédent, soit le troisième), ce qui nous donne : 172.18.3.64, correspond à l'adresse du prochain réseau. Donc 172.18.3.(64 – 1 = 63) correspond à l'adresse de diffusion (ou de broadcast).

La première adresse disponible est donc 172.18.1.(64 + 1 = 65).

La dernière adresse disponible est donc 172.18.3.(63 – 1 = 62).