

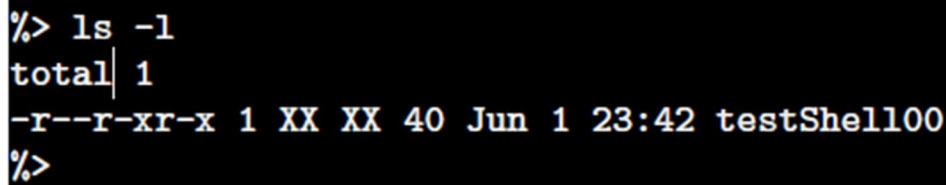
```

cd shell00          // navigue vers le répertoire shell00
mkdir ex01          // crée le répertoire ex01
cd ex01             // navigue vers le répertoire ex01

vim testShell00     // ouvre l'éditeur de texte vim pour afficher le contenu du fichier testShell00
                    // (le crée s'il n'existe pas encore)

<i>                 // (i = insert) : entre dans le mode insertion de vim (-- INSERT --) de vim

```



```

%> ls -l
total 1
-r--r-xr-x 1 XX XX 40 Jun 1 23:42 testShell00
%>

```

```

ls -l               // (-l = long listing format) : liste détaillée des fichiers du répertoire courant
                    // la première ligne (total x) affiche le nombre total de blocs
                    // (dépend de la taille des fichiers, nombre variable selon
                    // les différents systèmes d'exploitation)
                    // cela correspond à la somme totale de l'espace disque utilisé par les fichiers
                    // et répertoires dans le répertoire courant (exprimée en blocs de 512 octets)
                    // cela prend aussi en compte l'espace disque utilisé par la structure du
                    // répertoire elle-même ainsi que les métadonnées des fichiers
                    // (droits, dates de modification, etc)
                    // pour cela, ce nombre peut varier selon le système de fichier utilisé
                    // par le système d'exploitation

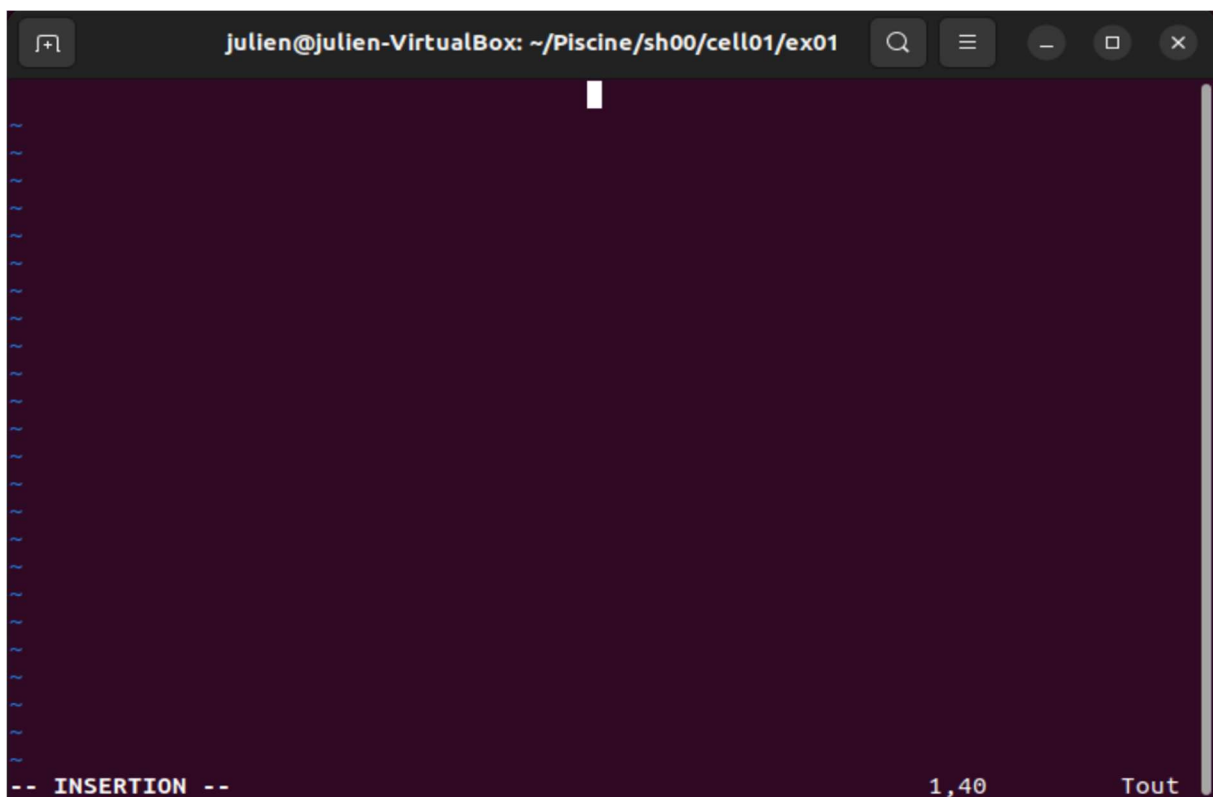
                    // puis tous les fichiers, liens et répertoires du dossier courant
                    // avec, pour chacun :
                    // le type : - pour un fichier régulier, d pour un répertoire,
                    // l pour un lien symbolique (symlink)
                    // - les droits de lecture, d'écriture et d'exécution pour l'utilisateur, le groupe
                    // et others (voir plus bas)
                    // le nombre de liens vers ce fichier (1 pour un fichier, 2 si ce fichier a (ou est)

```

```
// un lien vers un autre fichier)
// pour un répertoire, le nombre de liens est 2
// (un pour le répertoire lui-même (.) et un pour le répertoire parent (..))
// le nom de l'utilisateur propriétaire du fichier (ou du répertoire)
// le nom du groupe propriétaire du fichier (ou du répertoire)
// la taille du fichier en octet
// la date (en format Mois Jour Heure:Minutes) de la dernière modification du
// fichier
// le nom du fichier (ou du répertoire)
```

```
%> ls -l
total 1
-r--r-xr-x 1 XX XX 40 Jun 1 23:42 testShell00
%>
```

```
<space> x39 // insère 39 espaces dans le fichier pour que le fichier fasse 40 octets en taille
// vim insère un saut de ligne à la fin du fichier, ce qui compte pour 1 octet !
```



```
// 1,40 : indique que l'on est sur la 1ère ligne, à la 40ème colonne
// (39 caractères ont été tapés)
```

<escape> // sort du mode insertion de vim  
:wq // (: = commande vim) ; (w = write (sauvegarder)) ; (q = quit (quitter))  
// donc (:wq = commande vim pour sauvegarder et quitter vim)  
<enter> // valide la commande vim

```
%> ls -l  
total 1  
-r--r-xr-x 1 XX XX 40 Jun 1 23:42 testShell00  
%>
```

// le 1<sup>er</sup> tiret indique que testShell00 est un fichier

```
%> ls -l  
total 1  
-r--r-xr-x 1 XX XX 40 Jun 1 23:42 testShell00  
%>
```

// r-- : 1<sup>er</sup> ensemble de droits sur le fichier  
// correspond aux droits du « user » (le propriétaire du fichier) sur ce fichier  
// r : droit de lecture  
// w : droit d'écriture  
// x : droit d'exécution  
// ici, on a r-- :  
// le user a le droit de lire (r)  
// n'a pas le droit d'écrire  
// (w n'est pas indiqué, on a - à la place à la 2<sup>ème</sup> position  
// du 1<sup>er</sup> ensemble de droits sur le fichier)  
// ni le droit d'exécuter  
// (x n'est pas indiqué, on a - à la place à la 2<sup>ème</sup> position  
// du 1<sup>er</sup> ensemble de droits sur le fichier)  
  
// r-- se traduit en binaire par 100  
// (on remplace une lettre (r w ou x) par 1 et - par 0)  
// on traduit 100 en décimal  
// pour cela, on utilise la règle du 421 (YATZE !)

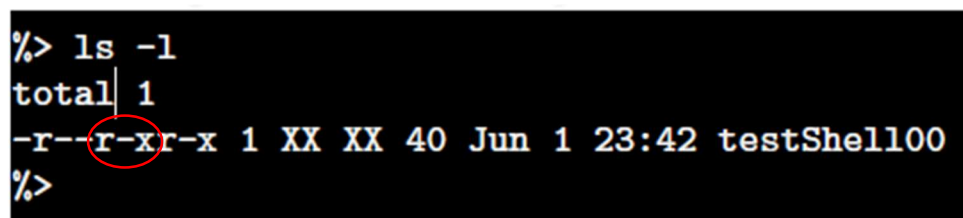
```
// (le dernier bit vaut 1, l'avant-dernier vaut 2, et l'avant avant-dernier vaut 4)

// on multiplie chaque bit par sa valeur :

// 1 0 0
// * * *
// 4 2 1
// = 4 0 0

// puis on additionne chaque chiffre :
// 4 + 0 + 0 = 4

// on répète l'opération avec le 2ème ensemble de droits sur le fichier
// qui correspond aux droits du « group »
// (du groupe d'utilisateurs) sur ce fichier
```



```
%> ls -l
total 1
-r--r-xr-x 1 XX XX 40 Jun 1 23:42 testShell00
%>
```

```
// ici, on a r-x :
// le group a le droit de le lire (r)
// n'a pas le droit de l'écrire
// (w n'est pas indiqué, on a - à la place à la 2ème position
// du 2ème ensemble de droits sur le fichier)
// a le droit de l'exécuter
```

```
// r-x se traduit en binaire par 101
```

```
// on traduit 101 en décimal
```

```
// 1 0 1
```

```
// * * *
```

```
// 4 2 1
```

```
// = 4 0 1
```

```
// puis on additionne chaque chiffre :
```

```
// 4 + 0 + 1 = 5
```

// on répète l'opération avec le 3<sup>ème</sup> ensemble de droits sur le fichier  
// qui correspond aux droits des « others » (de tous les autres) sur ce fichier

```
%> ls -l
total 1
-r--r-xr-x 1 XX XX 40 Jun 1 23:42 testShell00
%>
```

// ici, on a aussi r-x :  
// les others ont le droit de le lire (r)  
// n'ont pas le droit de l'écrire  
// (w n'est pas indiqué, on a - à la place à la 2<sup>ème</sup> position  
// du 2<sup>ème</sup> ensemble de droits sur le fichier)  
// ont le droit de l'exécuter

// r-x se traduit en binaire par 101  
// on traduit 101 en décimal  
// 1 0 1  
// \* \* \*  
// 4 2 1  
// = 4 0 1  
// puis on additionne chaque chiffre :  
// 4 + 0 + 1 = 5  
// on obtient donc 4, 5 et 5

chmod 455 testShell00 // (chmod = change file mode bits) : change les droits du fichier testShell00  
// en lui attribuant les droits indiqués (455)  
// (455 se traduit par r--r-xr-x)

```
%> ls -l
total 1
-r--r-xr-x 1 XX XX 40 Jun 1 23:42 testShell00
%>
```

```
touch -mt 06012342 testShell00 // touch permet aussi de mettre à jour la date de
                                // modification du fichier indiqué
                                // (-m = modification time) : permet de mettre à jour
                                // la date de modification du fichier (ou du répertoire)
                                // (-t = STAMP) : permet d'utiliser la date indiquée
                                // à la place de la date actuelle
                                // le format de la date qu'il faut indiquer est :
                                // MMDDhhmm
                                // avec :
                                // MM = mois (2 chiffres)
                                // DD = jour (2 chiffres)
                                // hh = heure (2 chiffres)
                                // mm = minutes (2 chiffres)
                                // Jun 1 23 :42 s'écrit donc 06012342 dans ce format
                                // ATTENTION : si la date indiquée est passée depuis plus de 6 mois,
                                // l'année (avec 4 chiffres) sera indiquée après un ls -l
                                // au lieu des heures et des minutes
```

```
tar -cf testShell00.tar testShell00 // compresse le fichier testShell00 dans l'archive testShell00.tar
// (-c = create) : crée l'archive
// -f testShell00.tar testShell00
// spécifie le nom de l'archive à créer (testShell00.tar)
// puis le nom du fichier à archiver (testShell00)
```

```
rm testShell00      // (rm = remove) : supprime le fichier testShell00  
                    // car le dossier de rendu ne doit contenir que le fichier spécifié !  
                    // (Files to turn in : testShell00.tar)
```