

- 1) Recherchez le nom d'un fichier quelconque de votre répertoire personnel à l'aide de la commande « find nomFichier »,
`find gx` (créez le fichier si besoin)
quel est l'intérêt d'une commande aussi simple par rapport à la commande `ls` ?
La commande permet normalement de faire une recherche récursive, à partir d'un dossier, pour le voir on peut déplacer le fichier dans un répertoire fils et refaire la recherche.
`mv gx /Music`
`find gx` (vous constaterez que la commande ne trouve pas le fichier)
- 2) Corrigez la commande précédente à l'aide de l'option adéquate.
`find -name gx` (la commande peut provoquer une erreur sur certaines machines vous pouvez corriger cela en délimitant le nom du fichier par des guillemets) `find -name "gx"`
- 3) Recherchez tous fichiers commençant par une lettre particulière.
`find -name "g*"`
- 4) A partir de quel répertoire la recherche se fait-elle par défaut ?
On constate que toutes les lignes commencent par « ./ » (point et slash) la recherche fait donc par défaut sur le répertoire courant et sa descendance.
Répétez la commande en spécifiant un répertoire différent,
`find /home -name "g*"`
que se passe-t-il ?
on remarque que certaines lignes retournées par `find` sont des erreurs « permission denied » ce qui est normale négligeable.
- 5) Empêchez l'affichage des erreurs.
`find /home -name "g*" 2> /dev/null`
on redirige la sortie d'erreur vers le fichier `/dev/null` qui est une sorte de fichier poubelle pour le texte qui efface tout ce qu'on lui redirige, cette astuce est pratique pour désencombrer le résultat des lignes d'erreurs inutiles.
- 6) Ajoutez à la commande précédente un critère de type répertoire « d » puis fichier ordinaire « f » puis lien symbolique « l »
`find /home -name "g*" -type d`
`find /home -name "g*" -type f`
`find /home -name "g*" -type l`
Remarque: si les commandes ne renvoient rien d'intéressant élargissez la recherche au répertoire racine /
- 7) Faite une nouvelle recherche par date de dernière modification (2 derniers jours)
`find . -mtime -2`
Par défaut le paramètre est spécifié en jours (nombre de jours X 24h), pour plus de détails, reportez-vous aux manuels de la commandes.
- 8) Faites une recherche à partir du répertoire racine, sans afficher les erreurs, des fichiers ordinaires dont le nom commence par « pass » et dont la taille est supérieure à 1KiO et inférieure à 4KiO et dont vous êtes le propriétaire.
`find / -type f -name "pass*" -size +2 -size -8 -user user01 2>/dev/null`
Conseil : construire la commande progressivement de sorte à repérer d'éventuelles erreurs, et faites attention à bien espacer les options et paramètres
Remarques : la taille est définie en nombre de blocs, sur beaucoup de machines le bloc est de 512KiO mais cela n'est pas une règle.
Le plus sert à spécifier que l'on veut des fichiers de taille supérieure à deux blocs, et le moins inférieurs à deux blocs,
L'option n'admet pas d'intervalles mais comme les options ont usage de critères (ou conditions) il est très possible d'utiliser deux fois la même option, ou encore des opérateurs logiques « and, or ,not » comme spécifié à l'arrière de la fiche.

Si la commande s'exécute correctement et ne renvoi rien enlevez des critères tels que l'user Extra : vous pouvez combiner le find avec la commande « wc -l » grâce à un tube (pipe) pour voir le nombre de résultats changer à chaque fois que l'on ajoute ou enlève des critères

- 9) Recommencez cette fois-ci avec les fichiers qui ne sont pas des répertoires.

```
find / ! -type d -name "pass*" -size +2 -size -8 -user user01 2>/dev/null
```

Si vous faites un pipe avec « wc » vous remarquerez que le nombre de fichier augmente (car on inclue en plus des fichiers d'autres type)

encore une fois si le résultat est infructueux vous pouvez réduire les critères pour les essais

- 10) Faites un ls -l de tous les fichiers de votre répertoire dont le nom commence par « g » à l'aide de l'option « -exec »

N.B. la commande suivant « -exec » doit recevoir comme paramètre (argument) la chaîne « {} » suivit de « \; »

```
find . -name "g*" -exec ls -l {} \;
```

les « {} » seront remplacés par les lignes renvoyés par la commande find, on obtiendra donc en résultat les informations détaillées sur chaque fichiers renvoyés par « ls -l »

le « ; » désigne la fin de la commande à exécuter, on y ajoute l'antislash « \ » pour le prendre comme un caractère spécial.

IMPORTANT !!! : Dans la commande qui suit (11), assurez-vous d'effectuer cette opération dans un répertoire sans importance (ou de test crée spécifiquement pour cet usage), cette commande va modifier les droits d'un grand nombre de fichier et de répertoires.

Ne lancez pas cette commande en tant que super utilisateur ou avec « sudo » sans savoir ce que vous faites.

Aussi faites attention à ne pas inverser les droits entre répertoires et fichiers ordinaires, cela vous empêchera d'accéder à vos propres répertoires.

- 11) Modifiez les droit d'accès des fichiers de votre répertoire en 666 et les répertoires en 777

```
find . \( -type f -exec chmod 664 {} \; \) , \( -type d -exec chmod 775 {} \; \)
```

On précède également les parenthèses par des antislashes. Il est possible d'ajouter l'opérateur or « -o » entre les deux ensembles à la place de la virgule.

Cette commande en particulier fonctionne sans les parenthèses car la commande s'exécute de gauche à droite. Dans une commande plus complexe les parenthèses peuvent s'avérer nécessaires.

- 12) Faites une copie du fichier /etc/passwd dans votre répertoire personnel,

```
cp /etc/passwd Gx
```

puis extrayez les lignes des utilisateurs :

- qui utilisent le shell bash

```
grep "bash" Gx
```

Le grep renvoi les lignes contenant le mot « bash » et donc les utilisateurs l'utilisant, cette commande n'est pas tout à fait juste car elle peut également renvoyer des lignes contenant le mot « bash » dans une autre champ (voir la question suivante).

- qui utilisent n'importe quel shell

```
grep "sh$" Gx
```

Sachant que tous les nom de shell se terminent par « sh » et que le shell se trouve dans le dernier champs de chaque ligne (et donc à la fin de chaque ligne)

le « \$ » désigne ici la fin de ligne

- dont le nom commence par la lettre « p »

```
grep "^p" Gx
```

L'accent circonflexe (chapeau) « ^ » désigne ici le début de ligne

- user01 jusqu'à 09

`grep "^user0[1-9]:" Gx`

Les crochets désigne ici un caractère dans l'ensemble donné

Remarque : au dos de la fiche vous trouverez deux tableaux de symboles différents, vous remarquerez que l'écriture des expressions régulières pour le shell et pour les commandes ne se fait pas de la même manière (les symboles changent)

- dont la ligne comporte les chaînes « uu » ou « uuu » (deux solutions)

`grep "u\{2,3\}" Gx` et `grep "uu \| uuu" Gx`

cette notation permet de définir un nombre d'occurrence pour le caractère « u »

- dont le nom commence par a, b ou c (deux solutions)

`grep "[abc]" Gx` et `grep "^[d-z]" Gx`

l'accent circonflexe n'a pas le même sens, entre crochets il désigne la négation

13) Recherchez les lignes contenant la chaîne de caractère « games » dans tous les fichiers de votre répertoire personnel,

Testez les options du `grep` comme vous savez le faire.

`grep -r "games" ~`

puis modifiez votre commande de sorte à obtenir le nombre ces lignes.

`grep -rc "games" ~`

La partie restante est laissé comme travail domestique, l'éditeur `vi` n'est pas très pratique en comparaison des éditeurs graphique, mais il peut s'avérer très utiles si on a pas d'accès graphique au système (dépannage, accès à distance, ...), cet éditeur à l'avantage d'exister sur toutes les distributions Unix/Linux, sachant l'importance des fichiers textuels pour les systèmes Unix, il vous servira tôt ou tard si vous travaillez avec n'importe quel système du genre.

14) Créez un fichier texte à l'aide de l'éditeur `vi`

`vi NomDeFichier`

et saisissez le texte suivant :

En informatique, le terme *shell* désigne un logiciel fournissant une interface à l'utilisateur pour des composantes d'un ensemble informatique plus grand.

Bien qu'il puisse aussi désigner une interface graphique, *shell* est plus généralement employé pour désigner un interpréteur de lignes de commandes pouvant accéder aux services et interagir avec le noyau d'un système d'exploitation. Dans le cas d'Ubuntu, un shell interagit avec le noyau Linux.

15) Testez les différentes commandes et profitez-en pour corriger les éventuelles erreurs.

16) Sauvegardez et fermez le fichier.