

1) redirige les résultat vers le fichier **tmp**.

par défaut l'entrée std des commandes est le clavier et la sortie std est l'écran exemple :

```
cat (sans rien)    clavier ⇐ cat ⇐ ecran
cat > fichier      clavier ⇐ cat ⇐ fichier
cat fichier        fichier ⇐ cat ⇐ ecran
```

2) La redirection > a pour effet de créer le fichier tmp si ce fichier existait déjà, il est tout simplement écrasé .

3) La redirection >>Le fichier **tmp** n'est pas écrasé, mais la sortie standard de la commande à été ajoutés en fin de fichier, à la suite du texte qui était déjà dans le fichier.

4) La commande doit donner le détail du fichier tmp et tmp2
le fichier tmp2 n'existe pas ce qui renvoi une ligne d'erreur
le fichier tmp quand à lui renvoi le résultat attendu

5) ls -l tmp tmp2 > /dev/null (affiche uniquement l'erreur)

6) ls -l tmp tmp2 2>/dev/null (affiche uniquement la sortie standard)

7) ls -l tmp tmp2 > /dev/null 2>&1 (n'affiche rien)

Vous pouvez également faire l'exemple suivant :

```
ls -l tmp tmp2 > tmp3 2> test4
```

```
cat tmp3
```

```
cat tmp4
```

pour voir que les fichiers tmp3 et tmp4 contiennent bien les sorties de ls.

8) sort (utilisée seule donne la main pour saisir du text)

Donner par exemple des chiffres dans le désordre sur chaque ligne, puis terminer avec Ctrl+D)

La commande trie le texte puis le renvoi

```
Clavier ⇐ tri (sort) ⇐ ecran
```

- ces entrées : un fichier

-ces sorties : le contenu trié.

9) sort < tmp redirection de l'entrée depuis le fichier tmp, par défaut le résultat sort sur la sortie standard, c'est à dire à l'écran

(ici nous changerons le contenu du fichier tmp, par kelke chose de plus simple comme des caractères dans le désordre sur chaque ligne, car le contenu du fichier etc est trop complexe)

```
tmp ⇐ sort ⇐ ecran
```

10) sort <tmp > tmp2 => (**>tmp2**) a pour effet d'envoyer le résultat dans un fichier tmp2, le résultat n'apparaît plus à l'écran, mais est sauvegardé dans un fichier.

```
tmp ⇐ sort ⇐ tmp2
```

11) trier le contenu du fichier tmp, et envoyer le résultat au fichier tmp2 ; la sortie de la première commande va être rediriger vers la 2eme commande qui sert à afficher les premières lignes triées du fichier tmp et ensuite supprimer le fichier intermédiaire tmp2.

« sort < tmp > tmp2 ; head -5 < tmp2 ; rm tmp2 »

Ici le fichier tmp2 est utilisé comme intermédiaire entre les deux commandes.

tmp $\Rightarrow$ sort $\Rightarrow$ tmp2 ---- tmp2 $\Rightarrow$ head $\Rightarrow$ écran

12) mv tmp2 tmp (cette commande devient donc obsolète pour la suite et est supprimée de la nouvelle fiche, cependant nous en profiterons pour noter que la commande mv utilisée comme ici nous permet de renommer tmp2 en tmp).

13) « **sort < tmp | head -5** » (tmp $\Rightarrow$ sort $\Rightarrow$ head $\Rightarrow$ écran) le résultat est le même que celui de la ligne de commandes précédente sauf qu'ici nous avons raccordé la sortie de « sort » directement avec l'entrée de « head » sans utiliser de fichier intermédiaire (tmp2).

14) l'idée ici est de remplacer le fichier tmp par la commande qui en est à l'origine par exemple « ls-l /etc » pour avoir quelque chose comme ça : **clavier (/etc) $\Rightarrow$ ls $\Rightarrow$ sort $\Rightarrow$ head $\Rightarrow$ écran**

Bien-sûr la commande « **sort < ls-l /etc | head -20** » est incorrecte ne fonctionnera pas, il faut en déduire que la redirection se fait uniquement entre *commande* et *fichier*

Quant au pipe il sert à connecter des *commandes* entre elles (et uniquement des commandes), il faut donc reformuler l'instruction comme dans 15)

15) **ls-l /etc | sort | head -20** (on appelle pipeline l'enchaînement de plusieurs pipes)

vous pouvez ici prendre une commande qui donnera un résultat plus clair que ls -l /etc (un simple ls suffira)

16) remarquer qu'il n'est pas impossible de combiner les deux (redirection et pipe) **ls-l /etc | sort | head -20 > tmp**

Le traitement peut donc se faire d'un fichier vers un autre en passant par plusieurs commandes. Même si pour nos exemples le traitement prends une fraction de secondes (la durée du traitement est donc imperceptible pour nous), si la durée de traitement prends du temps (un traitement plus lourd), il est possible de l'exécuter comme tâche de fond (en laissant la main pour lancer d'autres commandes) en ajoutant à la fin de la commande le symbole « & » :

ls-l /etc | sort | head -20 > tmp &

cela ne se ressentira pas dans cet exemple, mais vous devez commencer à comprendre que tout ce que fait la machine n'est perceptible que si on s'arrange pour qu'il le soit.

Pour ce faire vous pouvez faire l'essai avec la commande cat qui ne redonne pas la main jusqu'à son interruption manuelle (ctrl+c).

17) ici pas besoin de faire la démonstration, il faut juste comprendre qu'on peut insérer une liste de commandes dans un fichier texte (une sorte de script) et l'exécuter en lot.

Il faut aussi comprendre que les commandes ne sont en fait que des programmes que l'on manipule comme des légos (briques assemblables), et que le shell lui-même est un programme.

La solution à la question 17 est de transmettre à un fichier contenant des commandes à un autre shell et cela comme tâche de fond, grossièrement comme cela :

fichier_qui_contient_des_commandes $\Rightarrow$ bash & (équivalent graphiquement à lancer ces commandes une à une dans un second terminal ouvert, on peut continuer à utiliser le premier normalement sans attendre le résultat)