Corrigé du TP 3 Système

C. Pain-Barre

INFO - IUT Aix-en-Provence

version du 1/11/2012

Corrigé de l'exercice 1

Utiliser **cat** *pour afficher des fichiers, et pour en concaténer.*

1. **simpsons.txt**
2. **cat -n simpsons.txt**
3. **cat /etc/group**
4. *. . . pas besoin de corrigé pour cette question. . .*
5. **cat /etc/passwd**
6. **cat des_lignes.txt**
   cat acrostiche.txt
   cat des_lignes.txt acrostiche.txt

Corrigé de l'exercice 2

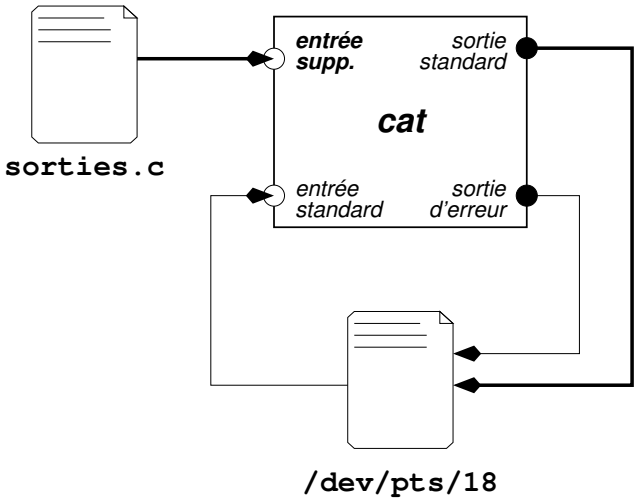
Utiliser **less** *pour paginer l'affichage.*

1. **less /etc/passwd**
2. **less a_decouper.txt ~/.bashrc**

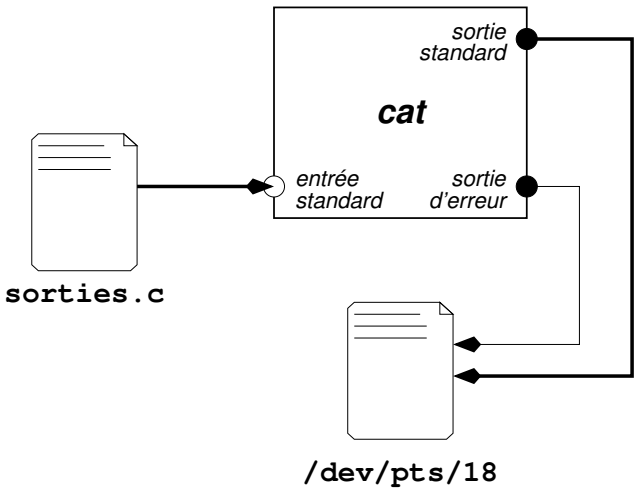
Corrigé de l'exercice 3

Étude des entrées-sorties, des terminaux et des redirections.

1. dans le corrigé, on suppose que c'est `/dev/pts/18`
2. **ls -l /dev/pts/18**
3. **ls -l /dev/pts**
4. **cat ~cpb/public/unix/sorties.c**
5. **cat** dispose des mêmes entrées-sorties que le shell, soit `/dev/pts/18`. Il a ouvert une entrée supplémentaire (en gras) pour lire le fichier, qu'il écrit (affiche) sur sa sortie standard (en gras). L'entrée standard et la sortie d'erreur n'ont pas été utilisées.



6. Dans ce cas, **cat** n'utilise que son entrée standard qui a été connectée au fichier par le shell.



7. `$ cat /etc/shadow`
   `cat: /etc/shadow: Permission non accordée`
   `$ ls -l /etc/shadow`
   `-rw-r----- 1 root shadow 64028 26 oct.  11:28 /etc/shadow`

   ➪ **cat** échoue. Avec `ls -l /etc/shadow` on peut voir que ce fichier appartient à `root` et est du groupe `shadow`. Les autres (dont nous faisons partie) n'ont aucun droit sur le fichier.

8. `$ cat < /etc/shadow`
   `-bash: /etc/shadow: Permission non accordée`

9. `cp ~cpb/public/unix/sorties ~/tp/tpunix`

10. . . . *pas besoin de corrigé pour cette question. . .*

11. (a) `$ ./sorties > fichierXX`
    `Cette ligne a été écrite sur la sortie d'erreur`
    `Cette ligne a été écrite sur la sortie d'erreur`
    `$`

    ➪ provoque la création du fichier `fichierXX` contenant le texte de la sortie standard, tandis que la sortie d'erreur reste affichée sur le terminal.

    (b) `$ ./sorties 2> fichierXX`
    `Cette ligne a été écrite sur la sortie standard`
    `Cette ligne a été écrite sur la sortie standard`
    `$`

➪ provoque l'écrasement du fichier `fichierXX` avec le texte de la sortie d'erreur, tandis que la sortie standard reste affichée sur le terminal.

(c)
```
$ ./sorties >& fichierXX
$
```

➪ provoque l'écrasement du fichier `fichierXX` contenant le texte des deux sorties et rien n'est affiché sur le terminal.

(d)
```
$ ./sorties >> fichierXX 2>&1
$
```

➪ ajoute le texte des deux sorties à la fin du fichier `fichierXX`. Rien n'est affiché sur le terminal.

12.
```
$ ls -l /dev/null
crw-rw-rw- 1 root root 1, 3 29 oct.  08:50 /dev/null
```

(a)
```
$ cat < /dev/null
$
```

➪ **cat** se termine aussitôt et n'affiche rien car ne lit que la fin de fichier.

(b)
```
$ ./sorties > /dev/null
Cette ligne a été écrite sur la sortie d'erreur
Cette ligne a été écrite sur la sortie d'erreur
```

➪ Le texte écrit sur la sortie standard disparaît dans le terminal fictif `/dev/null`. Le texte de la sortie d'erreur reste affiché sur le terminal.

```
$ cat /dev/null
$
```

➪ Le fichier `/dev/null` ne contient jamais rien quoi qu'on y écrive.

(c)
```
$ ./sorties 2> /dev/null
Cette ligne a été écrite sur la sortie standard
Cette ligne a été écrite sur la sortie standard
$
```

➪ Cette fois, on se débarrasse des messages d'erreur. Seul le texte de la sortie standard reste affiché sur le terminal.

(d)
```
$ ./sorties >& /dev/null
$
```

13. Il n'y a pas d'erreur car la redirection demande à bash de créer `fichier-nouveau.txt` (vide) **avant** de lancer l'exécution de **ls**

14. Si on regarde le contenu de `fichier-nouveau.txt`, on s'aperçoit qu'il avait une taille de 0 octets avant l'exécution de **ls** car il venait d'être créé par bash :

```
$ ls -l fichier-nouveau.txt > fichier-nouveau.txt
$ cat fichier-nouveau.txt
-rw-r--r--  1 cpb prof 0 sep 17 08:49 fichier-nouveau.txt
```

Durant son exécution, **ls** a écrit la ligne ci-dessus dans `fichier-nouveau.txt` qui n'est plus vide. On peut le vérifier en regardant la taille dans les détails sur `fichier-nouveau.txt` :

```
$ ls -l fichier-nouveau.txt
-rw-r--r--  1 cpb prof 58 sep 17 08:49 fichier-nouveau.txt
```

15. **`ls -l > liste.txt`**

    ➪ on pourra remarquer que `liste.txt` étant aussi créé **avant** l'exécution de **ls**, ses détails figureront dans `liste.txt` (mais avec une taille 0)...

16. **`cat essai1.txt essai2.txt > essai4.txt`**

17. `$` **`cat > cat.txt`**
    nnnppp Entrée
    CTRL-**D**
    `$`

18. Les messages d'erreur sont dus aux répertoires sur lesquels vous ne possédez pas le droit de lecture

19. **`ls ~cpb/* 2> /dev/null`**

    ➪ En détournant la sortie d'erreur de **ls** vers le terminal fictif "poubelle" `/dev/null`, les messages d'erreur disparaissent.

20. **`ls ~cpb/* 2> /dev/null > lisibles`**

Corrigé de l'exercice 4

Premiers pas avec les tubes (et les redirections) : combinaisons de **ls**, **less** *et* **cat**. *Il s'agit d'étudier les entrées-sorties des commandes présentes dans un tube.*

1. Un tube connecte les deux commandes. La sortie d'erreur de **sorties** n'est pas redirigée et reste le terminal. Elle est utilisée pour le message sur la sortie d'erreur. Sa sortie standard est redirigée par le tube en entrée de **cat** qui n'a pas d'argument, et qui écrit ce qu'elle lit en entrée sur sa sortie standard qui est aussi le terminal. La figure 1 illustre les entrées-sorties. Les traits fins soulignent les entrées-sorties qui ne sont pas effectivement utilisées.

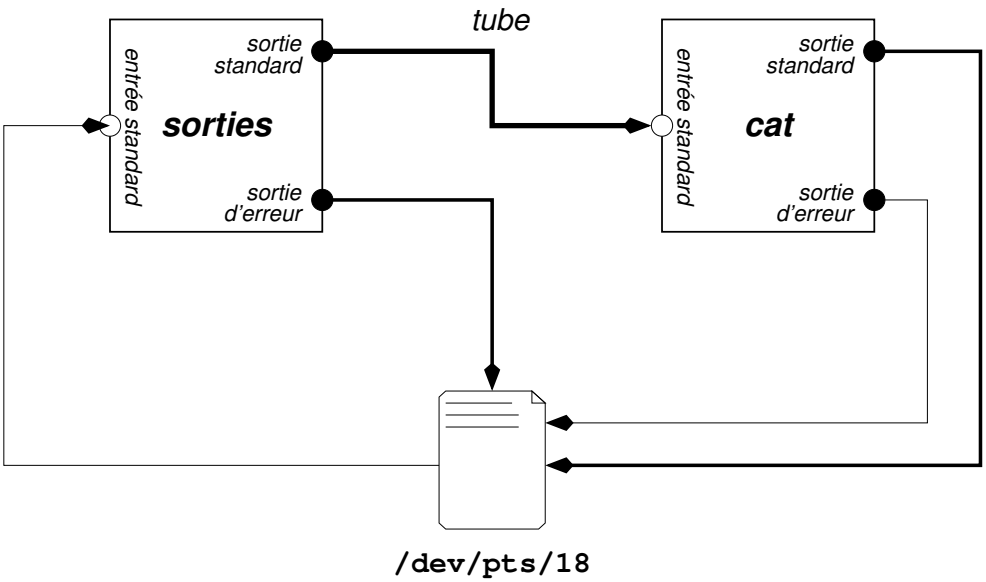


FIGURE 1 – Entrées-sorties de deux commandes connectées par un tube

Au final, **cat** et **sorties** se "disputent" l'écriture sur le terminal, ce qui peut produire un mélange du texte :

```
$ ./sorties | cat
Cette ligne a été écrite sur la sortie d'erreur
```

```
Cette ligne a été écrite sur la sortie d'erreur
Cette ligne a été écrite sur la sortie standard
Cette ligne a été écrite sur la sortie standard
```

ou un autre affichage.

2. `$ ./sorties 2>&1 | cat`
```
Cette ligne a été écrite sur la sortie standard
Cette ligne a été écrite sur la sortie d'erreur
Cette ligne a été écrite sur la sortie standard
Cette ligne a été écrite sur la sortie d'erreur
```
   ➪ **cat** récupère les deux sorties de **sorties** et elles sont bien écrites l'une après l'autre sur la sortie standard de **cat**.

3. `ls -l /home`
4. La sortie de **ls** et redirigée par le tube en entrée de **less** qui, n'ayant pas d'agument, pagine ce qu'elle lit sur son entrée (les messages normaux de **ls**).
5. `$ ./sorties | cat > fictube1`
```
Cette ligne a été écrite sur la sortie d'erreur
Cette ligne a été écrite sur la sortie d'erreur
$ cat fictube1
Cette ligne a été écrite sur la sortie standard
Cette ligne a été écrite sur la sortie standard
$
```
   ➪ le fichier `fictube1` est créé pour contenir la sortie de **cat**, c'est à dire les lignes «*Cette ligne a été écrite sur la sortie standard*».

6. `$ ./sorties | cat 2> fictube2`
```
Cette ligne a été écrite sur la sortie d'erreur
Cette ligne a été écrite sur la sortie d'erreur
Cette ligne a été écrite sur la sortie standard
Cette ligne a été écrite sur la sortie standard
$ cat fictube2
$
```
   ➪ le fichier `fictube2` est créé mais reste vide car **cat** n'écrit aucun message d'erreur. Le texte affiché est la sortie d'erreur de **sorties** (*Cette ligne a été écrite sur la sortie d'erreur*), et la sortie standard de **cat** (*Cette ligne a été écrite sur la sortie standard*). Ils peuvent apparaître mélangés.

7. `$ ./sorties 2> /dev/null | cat > fictube3`
```
$ cat fictube3
Cette ligne a été écrite sur la sortie standard
Cette ligne a été écrite sur la sortie standard
$
```
   ➪ Rien n'est affiché sur le terminal : les messages d'erreurs de **sorties** sont éliminés dans `/dev/null` et ses messages normaux sont repris par **cat** et finissent dans le fichier `fictube3`.

8. Le motif **`/home/*/p*`** est remplacé par l'ensemble des entrées commençant par **p** et contenues dans les sous-répertoires de `/home`, ce qui correspond aux entrées commençant par **p** présentes dans les répertoires d'accueil des utilisateurs d'allegro, et notamment `public` et `prive`, s'ils existent. **`ls -l`** est donc appliquée à chacune de ces entrées. Or, nous ne disposons pas forcément des droits suffisants pour afficher les détails sur toutes ces entrées, ou sur leur contenu pour celles qui correspondent aux répertoires, ce qui provoque l'affichage de messages d'erreur.

9. `ls -l /home/*/p* | less`
10. `ls -l /home/*/p* 2>&1 | less`
11. `ls -l /home/*/p* 2> /dev/null | less`
12. `ls -l /home/*/p* 2> /dev/null | cat | less`

Corrigé de l'exercice 5

Utiliser **wc** *pour réaliser des comptages divers, éventuellement combinée avec d'autres commandes*

1. `wc cigale.txt`
2. `wc -L cigale.txt`
3. `cat cigale.txt | wc -l`
4. `wc -l *.txt`
5. `cat *.txt | wc -l`
6. `ls /dev/pts/[0-9]* | wc -l`

Corrigé de l'exercice 6

Utilisation de **head** *et de* **tail**, *éventuellement combinées*

1. `cat des_lignes.txt`
2. `head -n 1 des_lignes.txt`
3. `head -n -2 des_lignes.txt`
4. `tail -n 2 des_lignes.txt`
5. `tail -n +6 des_lignes.txt`
6. `head -n 2 -q *.txt`
7. `head -n 1 -q *.txt > extremes ; tail -n 1 -q *.txt >> extremes`
8. `ls -l | head -n 1`
9. `tail -n +5 des_lignes.txt | head -n 3`

Corrigé de l'exercice 7

Utiliser **cut** *pour extraire des parties de lignes, éventuellement combinée à d'autres commandes*

1. (a) `cat /etc/passwd`
   (b) `cut -d: -f 1,3 /etc/passwd`
   (c) `cut -d: -f 1,3 /etc/passwd --output-delimiter='␣'`
2. (a) `ls -l` (si l'on est déjà dans `tpunix`)
   (b) En utilisant colonnes, on repère les colonnes des caractères :

```
...
-rw-r--r-- 1 toto toto 1168  1 nov.  11:20 amphigouri.txt
...
-rw-r--r-- 1 toto toto  552  1 nov.  11:20 cig.txt
$ ~cpb/public/bin/colonnes
000000000111111111122222222223333333333444444444455555555556...
123456789012345678901234567890123456789012345678901234567890...
```

Ici, la taille est comprise entre les caractères 24 et 27, et le nom du fichier commence au caractère 44, cela donne :

```
ls -l | tail -n +2 | cut -c 24-27,44- --output-delimiter='␣'
```

où l'on commence à supprimer la ligne `total` avec **tail** avant de ne retenir avec cut que les caractères qui nous intéressent.
On aurait pu se passer de **tail** et obtenir le même résultat avec :

```
ls -ld * | cut -c 24-27,44- --output-delimiter='␣'
```

(c) *... pas besoin de corrigé pour cette question...*

3. (a) *... pas besoin de corrigé pour cette question...*

   (b) **cat -n a_decouper.txt**

   (c) Sans les pointillés :
   head -n 29 a_decouper.txt | tail -n +9 | cut -c 25-111

4. (a) **du -k -s ~ | cut -f1**

   (b)
   ```
   $ cut -d';' -f3 ~cpb/public/unix/annuaire
   13013
   13008
   13013
   ...
   ```

   (c)
   ```
   $ cut -d';' -f3 ~cpb/public/unix/annuaire | sort | uniq
   13001
   13008
   13013
   ...
   ```

   (d)
   ```
   $ cut -d';' -f3 ~cpb/public/unix/annuaire | sort | uniq | wc -l
   5
   ```

Corrigé de l'exercice 8

Communication inter-terminaux avec les redirections/tubes. Dans cet exercice, on doit prêter une attention particulière aux entrées-sorties qui vont mélanger plusieurs terminaux.

1. *... pas besoin de corrigé pour cette question...*
2. **ssh allegro**
3. ```
   term1$ tty
   /dev/pts/23
   ```

   ➪ on suppose que **tty** a affiché `/dev/pts/23` dans **term1**

   ```
   term2$ tty
   /dev/pts/44
   ```
   ```

➪ et `/dev/pts/44` dans **term2**

4. `term2$` **echo bonjour > /dev/pts/23**
5. `term2$` **cat /etc/passwd > /dev/pts/23**
6. `term2$` **ls -l /users/etud | less > /dev/pts/23**

   ➪ la pagination est opérée par **less** dont la sortie est `/dev/pts/23` mais **less** se contrôle toujours via le terminal **term2** qui est son terminal de rattachement.

7. Si le voisin tente d'exécuter la commande : **echo 'un message' > /dev/pts/23** celle-ci échoue.
8. **chmod o+w /dev/pts/23**
   et du coup, **echo 'un message' > /dev/pts/23** tapé par le voisin réussit.
9. **chmod o-w /dev/pts/23**

ASCII Art

Corrigé de l'exercice 9

Utilisation des tubes en jouant avec **cowsay**, **cowthink**, **figlet**, **showfigfonts** *et* **boxes**.

1. `$` **figlet -f big 'Bonjour!' | cowsay -n**
2. `$` **figlet -f big 'Il fait un peu chaud...' | cowthink -n | boxes -d sunset**
3. `$` **ls ~ | figlet -f mini | boxes -d peek**
4. `$` **boxes -d c ~cpb/public/unix/sans-commentaire.cxx**
   > ~/tp/tpunix/tout-commentaire.cxx

   ➪ la commande précédente est à écrire sur une seule ligne

5. `$` **cat ~/tp/tpunix/cigale.txt | cowsay -n | boxes -d spring**
6. `$` **cat ~/tp/tpunix/{racine,amphigouri}.txt | boxes -d parchment**