Corrigé du TP 4 Système

C. Pain-Barre

INFO - IUT Aix-en-Provence

version du 13/11/2012

---

1 Manipulation des processus

Corrigé de l'exercice 1

Gérer des processus avec **jobs**, **ps**, **kill**, **killall** *et les caractères de contrôle.*

1. `pc:~$ ssh -X toto@allegro`
   `toto@allegro's password: ...`
   `Linux allegro...`
   `toto@allegro:~$`
2. `toto@allegro:~$ aterm`
   Une fenêtre terminal (de type aterm) apparaît alors.
3. `toto@allegro:~$ PS1='allegro2:\w\$ '`
   `allegro2:~$`
4. Parce que la commande **aterm** n'est pas terminée. Bash attend qu'elle soit terminée avant d'afficher le prompt et exécuter d'autres commandes.
5. Suite à la terminaison de **allegro2**, le bash sur **allegro1** reprend son exécution. Il lit alors la commande **ls** tapée précédemment et l'exécute.
6. Sur **allegro1** :
   `toto@allegro:~$ aterm`

   Sur **allegro2** :
   `toto@allegro:~$ PS1='\[\e]0;allegro2\a\]allegro2:\w\$ '`
   `allegro2:~$`
7. `toto@allegro:~$ aterm`
   CTRL-Z
   `[1]+  Stopped                 aterm`
   `toto@allegro:~$`

   ➪ le CTRL-Z stoppe l'*aterm* et débloque bash.
8. . . . *pas besoin de corrigé pour cette question. . .*
9. `toto@allegro:~$ jobs`
   `[1]+  Stopped                 aterm`
   `toto@allegro:~$`

   ➪ un seul job est encore en activité sur **allegro1** : c'est **aterm** qui est stoppé et qui porte le numéro de job 1
10. `toto@allegro:~$ jobs -l`
    `[1]+ 20430 Arrêté                 aterm`
    `toto@allegro:~$`

    ➪ Ce job numéro 1 pour **allegro1** est le processus de **PID** 20430 pour le système d'allegro.

11. Sur **allegro1** :

```
toto@allegro:~$ bg
[1]+ aterm &
toto@allegro:~$
```

➪ le shell indique qu'il fait reprendre l'exécution de **aterm** en parallèle

Sur **allegro2**, l'exécution de **tty** apparaît :

```
allegro2:~$ tty
/dev/pts/4
allegro2:~$
```

12. Comme prévu, l'exécution de **ls** est possible sur les deux fenêtres. Sur **allegro1** :

```
toto@allegro:~$ ls
prive  public  tp   ...
toto@allegro:~$
```

Sur **allegro2** :

```
allegro2:~$ ls
prive  public  tp   ...
allegro2:~$
```

13. Sur **allegro1** :

```
toto@allegro:~$ aterm&
[2] 21970
toto@allegro:~$
```

➪ il n'est pas nécessaire de séparer **aterm** et **&** par un blanc. On voit que le nouveau job porte le numéro 2 et a pour **PID** 21970.

Sur **allegro3** :

```
toto@allegro:~$ PS1='\[\e]0;allegro3\a\]allegro3:\w\$␣'
allegro3:~$
```

14.
```
allegro1:~$ ps
  PID TTY          TIME CMD
18131 pts/3    00:00:00 bash
20430 pts/3    00:00:00 aterm
21970 pts/3    00:00:00 aterm
23967 pts/3    00:00:00 ps
```

15.
```
allegro1:~$ ps l
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
0  8486 18131 18126  20   0   4904  2040 -      Ss   pts/3      0:00 -bash
0  8486 20430 18131  20   0  12856  3640 ?      S    pts/3      0:00 aterm
0  8486 20431 20430  20   0   4740  1940 -      Ss+  pts/4      0:00 bash
0  8486 21970 18131  20   0  12856  3632 ?      S    pts/3      0:00 aterm
0  8486 21971 21970  20   0   4740  1936 -      Ss+  pts/6      0:00 bash
0  8486 24172 18131  20   0   3492   844 -      R+   pts/3      0:00 ps l
```

➪ le bash de **allegro1** a le **PID** 18131. Il est bien le **PPID** des deux processus **aterm**

16. Sur **allegro2** :

```
allegro2:~$ tty
/dev/pts/4
```

Sur **allegro3** :

```
allegro3:~$ tty
/dev/pts/6
```

Sur **allegro1** :

```
allegro1:~$ ps lx
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY      TIME COMMAND
5  8486 18126 18122  20   0  11432  2016 ?      S    ?        0:00 sshd: toto...
0  8486 18131 18126  20   0   4904  2040 -      Ss   pts/3    0:00 -bash
0  8486 18441     1  20   0  12668  3116 -      Ssl  ?        0:00 /usr/lib/b...
0  8486 20430 18131  20   0  12856  3640 ?      S    pts/3    0:00 aterm
0  8486 20431 20430  20   0   4740  1940 -      Ss+  pts/4    0:00 bash
0  8486 21970 18131  20   0  12856  3632 ?      S    pts/3    0:00 aterm
0  8486 21971 21970  20   0   4740  1936 -      Ss+  pts/6    0:00 bash
0  8486 24309 18131  20   0   3492   844 -      R+   pts/3    0:00 ps lx
```

➪ le bash lancé par **allegro2** est rattaché au terminal `pts/4`. Son **PID** est `20431`. Le bash lancé par **allegro3** est rattaché au terminal `pts/6`. Son **PID** est `21971`.

17. Après avoir repéré le **PID** de **aterm3**, un signal gentil devrait suffire :

    ```
    allegro2:~$ kill pid-de-aterm3
    ```

18. ```
    allegro1:~$ aterm&
    [2] 26725
    allegro1:~$
    ```

19. ```
    allegro2:~$ ~cpb/public/unix/casse-pieds

    >(  Je suis le processus casse-pieds...
    >(  et je n'ai pas envie de me terminer de sitôt.

    ;)  Voyons si vous pouvez me faire dégager ...
    ...
    ```

    Le processus *casse-pieds* est insensible aux combinaisons CTRL-**C** et CTRL-**Z**, ainsi qu'aux signaux **TERM** et **INT**. En revanche, il est arrêté par le signal **KILL**. Pour le lui envoyer, il faut utiliser **ps lx** depuis **allegro1** (ou **allegro3**) pour repérer le **PID** du processus *casse-pieds*. Une fois son **PID** obtenu, on le tue depuis **allegro1** (ou **allegro3**) par :

    ```
    allegro1:~$ kill -KILL pid-de-casse-pieds
    ```

20. ```
    allegro1:~$ killall -u toto aterm
    [1]-  Complété                 aterm
    [2]+  Complété                 aterm
    allegro1:~$
    ```

Corrigé de l'exercice 2

Utilisation de **pstree** *et de* **top**.

1. *... pas besoin de corrigé pour cette question...*
2. `$ man pstree`
3. La sortie de **pstree** devrait ressembler à cet extrait :

   ```
   $ pstree
   init-+-NetworkManager-+-dhclient
        |                `-NetworkManager
        |-...
   ```
   ```

```
      |
      |-gconfd-2
      |-gdm3-+-gdm-simple-slav-+-Xorg
      |      |                 |-gdm-session-wor-+-x-session-manag-+-bluetooth-apple
      |      |                 |                 |                 |-evolution-alarm
      |      |                 |                 |                 |-gdu-notificatio
      |      |                 |                 |                 |-gnome-panel
      |      |                 |                 |                 |-gnome-power-man
      |      |                 |                 |                 |-gnome-terminal-+-bash---man---pager
      |      |                 |                 |                 |                |-bash---pstree
      |      |                 |                 |                 |                |-gnome-pty-helpe
      |      |                 |                 |                 |                |-...
      |      |                 |                 |                 |                `-gnome-terminal
      |      |                 |                 |                 |-gnome-volume-co
      |      |                 |                 |                 |-kerneloops-appl
      |      |                 |                 |                 |-metacity---metacity
      |      |                 |                 |                 |-nautilus
      |      |                 |                 |                 |-...
      |      |                 |                 |                 `-x-session-mana
      |      |                 |                 `-gdm-session-wo
      |      |                 `-gdm-simple-sla
      |      `-gdm3
...
```

➪ on voit qu'au sommet de la hiérarchie, il y a le processus **init** (`PID 1`). Tous les autres processus sont ses descendants directs ou indirects. Le processus **pstree** quant à lui se trouve tout à fait à droite, descendant direct d'un bash. Selon l'installation (et les circonstances), le **gnome-terminal** grand père de **pstree** n'est pas comme ici un descendant (indirect) de **gdm3** mais est un descendant direct de **init**.

4. `$ pstree -h`

```
init-+-NetworkManager-+-dhclient
     |                `-NetworkManager
     |-...
     |
     |-gconfd-2
     |-gdm3-+-gdm-simple-slav-+-Xorg
     |      |                 |-gdm-session-wor-+-x-session-manag-+-bluetooth-apple
     |      |                 |                 |                 |-evolution-alarm
     |      |                 |                 |                 |-gdu-notificatio
     |      |                 |                 |                 |-gnome-panel
     |      |                 |                 |                 |-gnome-power-man
     |      |                 |                 |                 |-gnome-terminal-+-bash---man---pager
     |      |                 |                 |                 |                |-bash---pstree
     |      |                 |                 |                 |                |-gnome-pty-helpe
     |      |                 |                 |                 |                |-...
     |      |                 |                 |                 |                `-gnome-terminal
     |      |                 |                 |                 |-gnome-volume-co
     |      |                 |                 |                 |-kerneloops-appl
     |      |                 |                 |                 |-metacity---metacity
     |      |                 |                 |                 |-nautilus
     |      |                 |                 |                 |-...
     |      |                 |                 |                 `-x-session-mana
     |      |                 |                 `-gdm-session-wo
     |      |                 `-gdm-simple-sla
     |      `-gdm3
...
```

5. `$ pstree toto`

```
...
x-session-manag-+-bluetooth-apple
                |-evolution-alarm
                |-gdu-notificatio
                |-gnome-panel
                |-gnome-power-man
                |-gnome-terminal-+-bash---man---pager
                |                |-bash---pstree
                |                |-gnome-pty-helpe
                |                |-...
                |                `-gnome-terminal
                |-gnome-volume-co
                |-kerneloops-appl
                |-metacity---metacity
                |-nautilus
                |-...
```

```
                `-x-session-mana
...
```

➪ pour l'utilisateur toto.

6. `$ top`

   ➪ l'affichage dépend de l'état du système

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

2 Utilisateurs et groupes

Corrigé de l'exercice 3

Gestion des groupes par **gpasswd**, **groups**, **newgrp** *et* **chgrp**

1. `allegro1$ id`
   `uid=1201(et1199) gid=1205(et1199) groupes=1205(et1199),1004(etud1)`
2. `allegro1$ groups`
   `et1199 etud1`
3. `allegro1$ less /etc/passwd`
   ```
   ...
   et1199:x:1201:1205:XX :/home/et1199:/bin/bash
   ...
   ```
4. `allegro1$ less /etc/group`
   ```
   etud1:x:1004:et1001,et1002,...,et1199,et1200
   et1199:x:1205:
   ```
5. `allegro1$ gpasswd -a et1196 et1199`
   `Ajout de l'utilisateur et1196 au groupe et1199`
   `allegro1$ gpasswd -a et1197 et1199`
   `Ajout de l'utilisateur et1197 au groupe et1199`
   `allegro1$ gpasswd -a et1198 et1199`
   `Ajout de l'utilisateur et1198 au groupe et1199`
   `allegro1$ ...`

   ➪ ajout des utilisateurs et1196, et1197 et 1198 au groupe et1199 (administré par et1199).

6. `allegro1$ grep et1199 /etc/group`
   `etud1:x:1004:et1001,et1002,...,et1197,et1198,et1199,et1200`
   `et1199:x:1205:et1196,et1197,et1198`

   ➪ et1196, et1197 et et1198 ont bien été ajoutés au groupe et1199

   ```
   allegro1$ groups et1196 et1197 et1198
   et1196 : et1196 etud1 et1199
   et1197 : et1197 etud1 et1199
   et1198 : et1198 etud1 et1199
   ```

7. `allegro1$ groups`
   `et1199 etud1`
   `allegro1$ groups et1199`
   `et1199 etud1 et1197`

➪ sur la connexion, l'utilisateur et1199 n'appartient qu'aux groupes et1199 et etud1, bien qu'il ait été ajouté dans le groupe et1197

8.
```
allegro1$ exit
Connection to allegro.iut.univ-aix.fr closed.
pc$ ssh et1199@allegro
...
allegro1$ groups
et1199 etud1 et1100 et1197 et1198
```

➪ *a priori*, et1199 a été ajouté aux groupes et1100, et1197 et et1198

9.
```
allegro1$ gpasswd -d et1196 et1199
Retrait de l'utilisateur et1196 du groupe et1199
allegro1$ gpasswd -d et1198 et1199
Retrait de l'utilisateur et1198 du groupe et1199
allegro1$ grep et1199 /etc/group
etud1:x:1004:et1001,et1002,...,et1197,et1198,et1199,et1200
et1199:x:1205:et1197
```

➪ il ne reste plus que et1197 comme membre (additionnel) de et1199

10.
```
allegro1$ groups
et1199 etud1 et1100 et1197 et1198
```

➪ même si et1199 a été enlevé des groupes et1100 et et1198 (après l'établissement de la connexion SSH)

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

11.
```
allegro1$ exit
Connection to allegro.iut.univ-aix.fr closed.
pc$ ssh et1199@allegro
...
allegro1$ groups
et1199 etud1 et1197
```

➪ où et1199 avait bien été enlevé des groupes et1100 et et1198

12.
```
allegro1$ gpasswd et1199
Changement du mot de passe pour le groupe et1199
Nouveau mot de passe : motdepasse
Nouveau mot de passe (pour vérification) : motdepasse
```

13.
```
allegro1$ newgrp et1197
```

➪ le mot de passe du groupe n'est pas demandé puisque et1199 appartient au groupe et1197

14.
```
$ touch vide.txt
```

15. On vérifie en tapant :

```
$ ls -l vide.txt
-rw-r--r-- 1 et1199 et1197 0 12 nov.  10:57 vide.txt
```

16.
```
$ chgrp et1199 vide.txt
```

17.
```
$ ls -l vide.txt
$ rm vide.txt
```

18.
```
$ exit
allegro1$
```

19. `allegro1$ newgrp et1100`
    `Mot de passe : motdepasse`
    `$`

    ➪ ne faisant pas partie du groupe et1100, le mot de passe du groupe est demandé.

20. `$ groups`
    `et1100 et1199 etud1 et1197`
21. `$ touch vide2.txt`
    `$ ls -l vide2.txt`
22. `$ rm vide2.txt`
23. `$ exit`
    `allegro1$`
24. `allegro1$ mkdir ~/groupe`
    `allegro1$ chmod 750 ~/groupe`
25. `allegro1$ cp ~/tp/tpunix/cigale.txt ~/groupe`
    `allegro1$ ls -l ~/groupe/cigale.txt`
    `allegro1$ chmod 660 ~/groupe/cigale.txt`
26. `allegro1$ cp ~/tp/tpunix/simpsons.txt ~/groupe`
    `allegro1$ ls -l ~/groupe/simpsons.txt`
    `allegro1$ chmod 640 ~/groupe/simpsons.txt`
27. `allegro1$ cp ~/tp/tpunix/{amphigouri,dates}.txt ~/groupe`
    `allegro1$ ls -l ~/groupe/{amphigouri,dates}.txt`
    `allegro1$ chmod 600 ~/groupe/{amphigouri,dates}.txt`
28. (a) *(non-membre)* `$ ls ~vous/groupe` *(doit échouer)*
        (non-membre) `$ cd ~vous/groupe` *(doit échouer)*
    (b) *(membre)* `$ ls ~vous/groupe` *(doit réussir)*
    (c) *(membre)* `$ cat ~vous/groupe/cigale.txt` *(doit réussir)*
    (d) *(membre)* `$ echo 'message du voisin' >> ~vous/groupe/cigale.txt` *(doit réussir)*

        ➪ la modification peut tout aussi bien se faire avec **vi** ou un autre moyen.

    (e) *(membre)* `$ echo 'message du voisin' >> ~vous/groupe/simpsons.txt` *(doit échouer)*
    (f) *(membre)* `$ cat ~vous/groupe/amphigouri.txt` *(doit échouer)*
29. `allegro1$ chmod g+w ~/groupe`
    (a) *(membre)* `$ echo salut > ~vous/groupe/simpsons.txt` *(doit échouer)*
        (membre) `$ echo salut >> ~vous/groupe/simpsons.txt` *(doit échouer)*
    (b) *(membre)* `$ rm ~vous/groupe/dates.txt` *(doit résusir)*
    (c) *(membre)* `$ vi ~vous/groupe/simpsons.txt`

        ➪ la sauvegarde de la modification doit réussir

    (d) *(membre)* `$ mv ~vous/groupe/amphigouri.txt ~vous/groupe/truc.txt`
    (e) *(membre)* `$ vi ~vous/groupe/truc.txt`

        ➪ la sauvegarde de la modification doit réussir

    (f) *(membre)* `$ cp ~/tp/tpunix/dates.txt ~vous/groupe`
        (membre) `$ chmod 640 ~vous/groupe/dates.txt`
    (g) `allegro1$ cat ~/groupe/dates.txt`
        échoue si vous n'appartenez pas au groupe du fichier.

(h) *(membre)* `$ chgrp` *groupe-commun* `~`*vous*`/groupe/dates.txt`

(i) `allegro1$ cat ~/groupe/dates.txt` *(doit réussir)*

(j) `allegro1$ echo saluit > ~/groupe/dates.txt` *(doit échouer)*
`allegro1$ echo saluit >> ~/groupe/dates.txt` *(doit échouer)*

30. `allegro1$ rm -rf ~/groupe`
31. . . . *pas besoin de corrigé pour cette question. . .*

Corrigé de l'exercice 4

Travail sur les utilisateurs et les groupes. Déduire les accès et droits à des fichiers/répertoires à partir de commandes exécutées.

	u2	u3	u4	u5	u6
r1	---	---	---	r-x	r-x
f1	---	---	---	rw-	rw-
r2	rwx	rwx	rwx	rwx	r--
f2	r-x	rw-	rw-	r-x	---
r3	r-x	---	r-x	---	---
f3	rwx	---	r--	---	---

3 Expressions régulières et utilitaires

Corrigé de l'exercice 5

Recherche des lignes de fichiers contenant certaines chaînes de caractères avec **`grep`**

1. Depuis `tpunix` :
   `cp ~cpb/public/unix/fruit.price .`
2. **`cat fruit.price`**
3. (a) **`grep berries fruit.price`**

   ➪ ici, il n'est pas nécessaire de protéger les caractères de l'expression régulière à rechercher. Par la suite, on les protégera.

   (b) **`grep -i 'appLES' fruit.price`**

   ➪ bien entendu, si on ignore la casse, on peut tout aussi bien écrire **`apples`** ou **`APPLES`** plutôt que **`appLES`**

   (c) **`grep -v 'apples' fruit.price`**

   (d) **`grep -w 'apples' fruit.price`**

   (e) **`grep '^s' fruit.price`**

   (f) **`grep '^[aeiouy]' fruit.price`**

   (g) **`grep '^[pa-g]' fruit.price`** ou **`grep '^[a-gp]' fruit.price`**

   (h) **`grep '^[^aeiouy]' fruit.price`**

➪ on peut aussi simplement inverser la recherche des lignes commençant par une voyelle avec l'option **-v** de **grep**, ce qui donne :

```
grep -v '^[aeiouy]' fruit.price
```

(i) `grep '^[^a-m]' fruit.price` ou `grep -v '^[a-m]' fruit.price`

(j) `grep '^[a-z]\{6\}' fruit.price`

(k) `grep '^[a-z]\{5\}␣' fruit.price`

➪ on recherche ici les lignes commençant par 5 lettres suivies d'un espace. On a utilisé le modificateur accolades mais on aurait pu écrire :

```
grep '^[a-z][a-z][a-z][a-z][a-z]␣' fruit.price
ou encore grep '^[^␣][^␣][^␣][^␣][^␣]␣' fruit.price
ou encore grep '^[^␣]\{5\}␣' fruit.price
```

(l) `grep '^p.....␣' fruit.price`

➪ Après le **p**, il peut y avoir 6 caractères quelconques, mais suivis d'un espace. Une autre solution est d'utiliser les modificateurs pour indiquer qu'il faut entre 0 et 6 lettres après le **p**, suivies d'un espace, ce qui donne :

```
grep '^p[a-z]\{0,6\}␣' fruit.price
voire plus simplement grep '^p.\{0,6\}␣' fruit.price
```

(m) `grep '79$' fruit.price`

(n) `grep '[^9]$' fruit.price` ou `grep -v '9$' fruit.price`

(o) `grep '[^3678]9$' fruit.price`

(p) `grep -v '[037]9$' fruit.price`

(q) `grep '␣\.' fruit.price`

➪ il faut que le point du prix soit précédé d'un espace

(r) `grep '[␣0]\.' fruit.price`

➪ il est inférieur à 1 euro si le point du prix est précédé d'un espace ou de **0**

(s) `grep '^[a-g].*[␣0]\.' fruit.price`

➪ On utilise `.*` pour indiquer qu'il peut y avoir n'importe quoi entre la première lettre et le prix. On aurait pu aussi combiner deux **grep** dans un tube :

```
grep '^[a-g]' fruit.price | grep '[␣0]\.'
```

Le premier, pour ne garder de `fruit.price` que les lignes qui commencent par une lettre comprise entre **a** et **g** ; le second, pour ne garder de ce qu'écrit le premier (et pas de `fruit.price` ! ! !) que les lignes dont le prix est inférieur à 1 euro.

(t) `grep '␣1\.[0-4]' fruit.price`

(u) `grep '^\([a-e]\|.[ae]\)' fruit.price` ou `grep '^[a-e]\|^.[ae]' fruit.price"`

➪ si on n'utilise pas les parenthèses comme dans la deuxième écriture, il faut penser à faire précéder la deuxième expression régulière par **^**.

Corrigé de l'exercice 6

Utilisation de **sed** *(en particulier combinée à d'autres commandes)*

1. (a) `sed -e 'y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/' *.txt`

   (b) `ls *.txt | sed -e 'y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/'`

   > ⓘ Pour cette question (et la précédente), il existe d'autres méthodes dont certaines sont beaucoup plus simples. Notamment, on peut utiliser la séquence spéciale[1] **\u** dans la chaîne de remplacement de la commande **s** de **sed** et écrire :
   >
   > `ls *.txt | sed -e 's/\([a-z]\)/\u\1/g'`
   > ou `ls *.txt | sed -e 's/\(.\)/\u\1/g'`
   > ou encore `ls *.txt | sed -e 's/./\u&/g'`
   >
   > On peut aussi utiliser la commande **tr** (non présente dans le poly) :
   > `ls *.txt | tr 'abcdefghijklmnopqrstuvwxyz' 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'`
   > ce qui, dans ce cas, n'est pas forcément plus simple...

   (c) `ls|sed -e '/\.txt$/y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/'`

   ➪ il faut penser à précéder le **.** de l'extension **.txt** par un backslash. De plus l'extension est placée à la fin de la ligne (d'où le **$**).

   (d) `sed -e 's/^/␣␣␣␣␣/' cigale.txt`

   (e) `sed -e 's/^␣*//' decale.txt`

   (f) Il y a plusieurs façons de faire. Soit on mémorise ce qu'on veut garder pour le réécrire (ou le réutiliser différemment) et on élimine le reste :
   `ls -l *.txt | sed -e 's/^\([^␣]*\).*\(␣[^␣]*\)$/\1\2/'`
   soit on élimine uniquement ce qu'on ne veut pas (en le remplaçant par un espace) en se disant que ce qu'on veut éliminer est entre le premier et le dernier espace de la ligne. En utilisant le fait que **.*** correspond à la plus grande chaîne possible, on peut simplement écrire :
   `ls -l *.txt | sed -e 's/␣.*␣/␣/'`

   (g) `ls -l | grep '^-' | sed -e 's/^.\([^␣]*\).*\(␣[^␣]*\)$/\1\2/'`

   > ⓘ on peut aussi utiliser avantageusement l'option **-n** de **sed** et l'option **p** de sa commande **s**, ce qui permet de ne pas utiliser **grep** :
   >
   > `ls *.txt | sed -n -e '/^-/s/^.\([^␣]*\).*\(␣[^␣]*\)$/\1\2/p'`

2. on peut créer un script `saute.sed` contenant simplement :

   `s/\.\+/&\n/g'`

   puis taper `sed -f saute.sed amphigouri.txt`
   Mais si **\n** n'est pas interprétée par cette version de **sed**, le script doit contenir les 2 lignes suivantes :

   ```
   s/\.\+/&\
   /g
   ```

1. Cette séquence est aussi valide dans **vi**. Pour plus d'informations et découvrir d'autres possibilités, demander dans **vi** l'affichage de l'aide en tapant `:help sub-replace-special`

Sans script, on obtient la même chose avec

```
$ sed -e 's/\.\+/&\n/g' amphigouri.txt
```

et si **\n** n'est pas interprétée, il faut taper les deux lignes suivantes, où le retour à la ligne est protégé par un backslash :

```
$ sed -e 's/\.\+/&\
/g' amphigouri.txt
```

Corrigé de l'exercice 7

Combinaison de **cut** *et* **sed**

1. *. . . pas besoin de corrigé pour cette question. . .*
2. `ps lx | sed -e 's/␣\+/␣/g' | cut -d'␣' -f 2,3,13`

   ➪ on remplace les suites d'espaces par un seul espace, ce qui permet d'utiliser l'option **-f** de **cut** en utilisant l'espace comme séparateur de champ.

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