TP commandes unix

É. Carry, J.-M Friedt

13 septembre 2016

Parmi les nombreux ouvrages décrivant les commandes unix et l'utilisation du shell, les livres édités par Tim O'Reilly sont tous excellents et méritent à être consultés. Pour le sujet qui nous concerne ici, *Classic Shell Scripting* de A. Robbins & N.H.F. Beebe (disponible à `http://proyuacouart.wikispaces.com/file/view/Classic_Shell_Scripting.pdf`) sera une référence utile.

1 Arborescence

L'arborescence de GNU/Linux et comment s'y retrouver : dans l'ordre de priorité :

— `/` est la racine de l'arborescence. Le séparateur entre répertoire sera toujours le `/`.
— `/home` contient les répertoires des utilisateurs. Unix, et Linux en particulier, impose un confinement strict des fichiers de chaque utilisateur dans son répertoire. Seul l'administrateur (`root`) peut accéder aux fichiers du système en écriture : personne ne travaille jamais comme administrateur.
— `/usr` contient les programmes locaux à un ordinateur : applications, entêtes de compilation (`/usr/include`), sources du système (`/usr/src`)
— `/lib` contient les bibliothèques du système d'exploitation
— `/proc` contient des pseudo fichiers fournissant des informations sur la configuration et l'état du matériel et du système d'exploitation. Par exemple, `cat /proc/cpuinfo`.
— `/etc` contient les fichiers de configuration du système ou la configuration par défaut des logiciels utilisés par les utilisateurs en l'absence de fichier de configuration locale (par exemple `cat /etc/profile`).
— `/bin` contient les outils du système accessibles aux utilisateurs et `/sbin` contient les outils du systèmes qu'*a priori* seul l'administrateur a à utiliser.
— . (un point) est le répertoire courant
— .. (deux points) est le répertoire au-dessus du répertoire courant et est utilisé pour remonter dans l'arborescence

Les commandes de base :

— `ls` : list, afficher le contenu d'un répertoire
— `mv` : move, déplacer ou renommer un fichier. `mv source dest`
— `cd` : change directory, se déplacer dans l'arborescence. Pour remonter dans l'arborescence : `cd ..`
— `rm` : remove, effacer un fichier
— `cp` : copy, copier un fichier. `cp source dest` Afin de copier un répertoire et son contenu, `cp -r repertoire destination`.
— `cat` : afficher le contenu d'un fichier. `cat fichier`
— `man`, manuel, est la commande la plus importante sous Unix, qui donne la liste des options et le mode d'emploi de chaque commande.

Afin de remonter dans la fenêtre du terminal, utiliser SHIFT+PgUp. Toute commande est lancée en tâche de fond lorsqu'elle est terminée par `&`.

L'interface graphique est une surcouche qui ralentit le système d'exploitation et occupe des ressources, mais peut parfois faciliter la vie en proposant de travailler dans plusieurs fenêtres simultanément. Elle se lance au moyen de `startx`.

2 Commandes simples

2.1 man

La commande la plus importante, et au pire la seule à connaître, est `man` pour accéder aux pages de documentation. `man command` informe de l'utilisation d'une commande, ses arguments et ses options. Les pages de manuel sont organisées par sections, avec l'organisation décrite dans ... `man man`, et reproduite ci-dessous

```
1   Executable programs or shell commands
2   System calls (functions provided by the kernel)
3   Library calls (functions within program libraries)
4   Special files (usually found in /dev)
```

```
5   File formats and conventions eg /etc/passwd
6   Games
7   Miscellaneous  (including  macro  packages and convenâĂŘ
          tions), e.g. man(7), groff(7)
8   System administration commands (usually only for root)
```

Observer la différence entre `man 2 open` **et** `man 1 open`**. Commenter**

Si une commande n'est pas connue, `man -k mot_cle` recherche l'occurence du mot clé dans la description des pages de manuel et facilite l'identification de la commande associée.

Imaginons que nous voulions trouver la commande en C qui nous permet de changer l'ordre des octets dans un mot de plus de 8 bits : taper la commande `man -k order` **et identifier la commande appropriée.**

2.2 grep

`grep` recherche l'occurence d'un motif dans une chaîne de caractères. Elle est très utile pour *filtrer* la masse d'informations issues des autres commandes que nous verrons plus bas, et s'utilise donc quotidiennement pour rendre le résultat d'une recherche digeste.

Nous désirons connaître les utilisateurs ayant accès aux ports série du PC, et donc appartenant au group *dialout*.

`grep dialout /etc/group`

`grep` recherche la présence d'une chaîne de caractères dans l'argument qui lui est fourni, par exemple un fichier. Il sait aussi recherche récursivement dans une arborescence avec l'option `-r`

Nous avons vu plus haut que la commande `htons` **inverse les octets d'un mot de 16 bits. Nous désirons connaître l'implémentation de cette commande : trouver l'occurence de cette commande dans le répertoire des fichiers d'entête et ses sous-répertoires** `/usr/include/linux`

`grep` propose une multitude d'options qui s'apprennent avec le temps : `-i` pour ne pas faire dépendre la recherche de majuscule/minuscule, `-v` pour rejeter au lieu d'accepter la correspondance du motif, `-A` et `-B` pour afficher les lignes avant et après celle correspondant au motif.

2.3 dmesg

Les messages du système sont contenus dans `/var/log/messages`, qui n'est accessible en lecture qu'à l'administrateur (root). Il est cependant possible pour l'utilisateur d'accéder à ces informations par `dmesg`, en particulier pour connaître le point de montage d'une clé USB par exemple.

```
[ 2804.721062] sd 2:0:0:0: [sdb] No Caching mode page found
[ 2804.721075] sd 2:0:0:0: [sdb] Assuming drive cache: write through
[ 2804.721966]  sdb: sdb1
[ 2804.727402] sd 2:0:0:0: [sdb] No Caching mode page found
[ 2804.727418] sd 2:0:0:0: [sdb] Assuming drive cache: write through
[ 2804.727428] sd 2:0:0:0: [sdb] Attached SCSI removable disk
```

Analyser cette séquence d'informations : quel est le format de la clé et comment accéder aux informations qui y sont stockées?

2.4 Le tuyau (*pipe*)

Les diverses commandes que nous étudions prennent pour argument un nom de fichier sur lequel appliquer le traitement. Un nom de fichier particulier est l'entrée standard – *stdin* – qui est le choix par défaut si aucun nom de fichier n'est fourni. L'entrée standard peut être connectée au clavier, et dans ce cas le traitement se fait sur les lettres tapées par l'utilisateur, une fonction peu utile. Plus intéressant, il est possible de connecter l'entrée standard à la sortie – *stdout* – de la commande précédente, par le tuyau (*pipe*) symbolisé par le caractère `|`. Ainsi, `dmesg | grep sd` propose toutes les opérations sur les supports de stockage au format compatible SCSI dans les messages du système

```
[ 1565.340777] sd 0:0:0:0: [sda]
[ 1565.340790] sd 0:0:0:0: [sda] CDB:
[ 1565.340813] end_request: I/O error, dev sda, sector 13813373
[ 1569.028648] sd 0:0:0:0: [sda] Unhandled sense code
[ 1569.028655] sd 0:0:0:0: [sda]
[ 1569.028664] sd 0:0:0:0: [sda]
[ 1569.028712] sd 0:0:0:0: [sda]
[ 1569.028725] sd 0:0:0:0: [sda] CDB:
[ 1569.028747] end_request: I/O error, dev sda, sector 13813373
```

Le *pipe* est une fonction puissante qui permet de cascader les opérations, un point clé dans le traitement de chaînes de caractères tel que nous le verrons ci-dessous.

2.5 wget

Internet est devenu incontournable, mais l'accès aux informations qui y sont stockées est freiné par les interfaces graphiques aussi lentes qu'inconfortables pour rendre la recherche des données systématiques. Parmi les multitudes de protocoles permettant d'accéder aux données accessible par internet, HTTP [1] décrit l'accès au web. HTTP est implémenté dans la commande `wget` qui permet depuis la ligne de commande d'accéder à un site.

1. **Accéder à la page web** `http://jmfriedt.free.fr`**. Sachant que la sortie de la commande peut être la sortie standard si nous le demandons par** `-O -`**, filtrer uniquement les énoncés de documents relatifs au Master 1 d'électronique.**
2. **Récupérer toutes les images contenues dans le répertoire accessible sur le web** `http://jmfriedt.sequanux.org/130929_kvad/` **: étudier les options de** `wget` **nécessaires à atteindre ce résultat.**

2.6 find

`find` est une commande à la syntaxe un peu complexe mais fort utile pour trouver un ou des fichier(s) dans une arborescence.

Recherche d'un fichier : `find` (locate) qui prend comme argument le répertoire de départ de la recherche, la nature de la recherche (nom du fichier, date de création, permission ...), le motif à rechercher, et l'opération à effectuer.

Exemple : on trouve l'emplacement de `stdio.h` par `find /usr/include/ -name stdio.h -print`. L'action `-print` est celle effectuée par défaut si aucune action n'est précisée.

Rechercher tous les fichiers liés au microprocesseur 32u4 dans le répertoire des fichiers d'entête de `avr-gcc`**, à savoir** `/usr/lib/avr`

Une option très puissante de `find` est l'exécution d'une commande sur le résultat de la recherche, par l'action `-exec`. Dans ce cas, l'argument est passé sous forme de la chaîne `{}` et la commande se termine par `\;`

Exemple : faire défiler toutes les photos d'un répertoire au moyen de l'affichage par la commande `display` de ImageMagick

```
find . -name '*.jpg' -exec display -delay 10 {} \;
```

Une commande fort utile pour passer le résultat d'une commande comme argument de la commande suivante est `xargs`. Ainsi, au lieu d'utiliser l'option `-exec` de `find`, nous pouvons passer le résultat de `find` comme argument de la commande suivante par `find . -name "*.c" | xargs ls -l` ou pour chercher les fichiers faisant appel aux entêtes des bibliothèques pour le microcontrôleur AVR : `find . -name '*.c' | xargs grep 'avr'`

2.7 cut

Au sein d'une ligne, nous voulons sélectionner (couper) certaines colonnes selon un motif – colonnes d'abscisses connues, ou de numéro de champ connu selon un séparateur connu par exemple.

Couper une/plusieurs colonnes d'un fichier selon sa position en nombre de caractères : `cut -c debut-fin`

Couper une/plusieurs colonnes d'un fichier selon sa position en indice du champ (`-f`) en choisissant un certain délimiteur (`-d`) : `cut -d\  -f 1-3`

Application : soit le fichier de trames GPS suivant

```
$GPGSA,A,3,02,24,12,10,,,,,,,,,3.2,3.0,1.0*3
$GPRMC,182653.000,A,4714.9166,N,00601.2024,E,1.16,84.77,090709,,*38
$GPGGA,182654.000,4714.9176,N,00601.2037,E,1,07,2.1,240.2,M,48.0,M,,0000*52
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,,3.2,2.1,2.4*3A
$GPRMC,182654.000,A,4714.9176,N,00601.2037,E,1.56,62.01,090709,,*31
$GPGGA,182655.000,4714.9182,N,00601.2047,E,1,07,2.1,242.0,M,48.0,M,,0000*5F
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,,3.2,2.1,2.4*3A
$GPRMC,182655.000,A,4714.9182,N,00601.2047,E,1.85,58.42,090709,,*3C
$GPGGA,182656.000,471.9189,N,00601.2055,E,17,2.1,244.8,M,48.0,M,,0000*5A
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,,3.2,2.1,2.4*3A
$GPGSV,3,1,11,29,85,043,20,31,58,275,20,30,51,103,25,24,48,127,25*7E
$GPGSV,3,2,11,21,28,172,,12,21,105,24,02,13,039,30,16,12,291,*70
$GPGSV,3,3,11,23,05,334,,10,05,078,28,14,02,219,*45
$GPRMC,182656.000,A,4714.9189,N,00601.2055,E,1.28,39.53,090709,,*37
$GPGGA,182657.000,4714.9195,N,00601.2057,E,1,07,2.1,245.3,M,48.0,M,,0000*5E
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,,3.2,2.1,2.4*3A
$GPRMC,182657.000,A,4714.9195,N,00601.20,E,1.13,18.42,090709,,*32
$GPGGA,182658.000,4714.9202,N,00601.2061,E,1,07,2.1,246.8,M,48.0,M,,0000*51
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,,3.2,2.1,2.4*3A
$GPRMC,182658.000,A,4714.9202,N,00601.2061,E,1.58,41.94,090709,,*3D
$GPGGA,182659.000,4714.9206,N,00601.2065,E,1,07,2.1,247.5,M,48.0,M,,0000*5C
```

1. RFC7230 à `https://tools.ietf.org/html/rfc7230`

```
$GPGSA,A,3,02,29,24,12,31,10,30,,,,,,3.2,2.1,2.4*3A
$GPRMC,182659.000,A,4714.9206,N,00601.2065,E,1.36,44.79,090709,,*32
$GPGGA,182700.000,4714.9216,N,00601.2072,E,1,07,2.1,250.0,M,48.0,M,,0000*55
```

Proposer la commande qui ne garde que les lignes contenant GPGGA. Étendre la commande pour ne garder que les colonnes contenant l'horaire, la latitude et la longitude.

2.8 head et tail

L'affichage de `dmesg` prend beaucoup de temps dans un terminal car tous les messages depuis le démarrage de l'ordinateur sont affichés – un message très long lorsque l'ordinateur n'a pas été éteint depuis plusieurs mois. Nous ne nous intéressons généralement qu'aux dernières informations les plus récentes, donc la fin du fichier `/var/log/messages`.

Nous pouvons couper une/plusieurs lignes d'une fichier à partir du début ou de la fin avec `head` et `tail` respectivement. Application : `dmesg | tail`

Afficher les 3 dernières lignes de la séquence NMEA proposée auparavant. Les trois premières.

2.9 regexp

Les arguments des commandes que nous avons vu auparavant, et en particulier `grep`, peuvent être de simples chaînes des caractères, ou des motifs décrivant des classes de chaînes de caractères. Ainsi, `[a-z]` indique toutes les lettres minuscules tandis que `[a-zA-Z]` décrit tout l'alphabet. Certains caractères spéciaux indiquent une position dans la ligne : `^` indique le début de ligne, et `$` la fin de ligne. Le symbole `.` indique "n'importe quel caractère".

Étendre la commande de la section 2.7 pour ne garder que les informations valides et non-corrompues, à savoir une partie entière de la latitude contenant 4 chiffres. Modifier la commande pour garantir que tous ces caractères sont bien des chiffres.

2.10 sed

Une fois que nous savons trouver des informations sur internet, nous ne pourrons nous empêcher d'y rechercher toutes les données et les mettre en forme exploitables par nos applications, voir lisibles par un humain [2]. Pour ce faire, nous aurons besoin de retirer des chaînes de caractère sans intérêt (par exemple le formatage d'une page web) ou remplacer certaines chaînes de caractères par d'autres. L'outil pour ce faire est `sed`.

1. **Effectuer une recherche pour un horaire et un tarif entre deux destinations françaises sur le site** `www.bahn.com/i/view/GBR/en/`
2. **Analyser le format de l'url et en déduire comment introduire une gare de destination dans la requête. Proposer une solution pour lire la liste des gares de destination dans un fichier.**
3. **analyser le code source de la page HTML résultante (CTRL-U dans firefox, ou effectuer la requête de la page web par** `wget` **tel que vu auparavant).**
4. **extraire les durées de trajets proposés. On notera que pour remplacer un retour chariot par un espace, nous devons passer par la commande** `tr` **puisque** `sed` **travaille toujours ligne par ligne. Ainsi,** `tr '\n' ' '` **remplace le retour chariot de fin de ligne par un espace, et concatène donc les deux lignes.**

2.11 Cas pratique : tracer une courbe récupérée sur un oscilloscope

Un oscilloscope LeCroy Waverunner renvoie, en réponse à la requête `C2:INSPECT? DATA_ARRAY_1,WORD`, le contenu de sa voie numéro 2 sous la forme

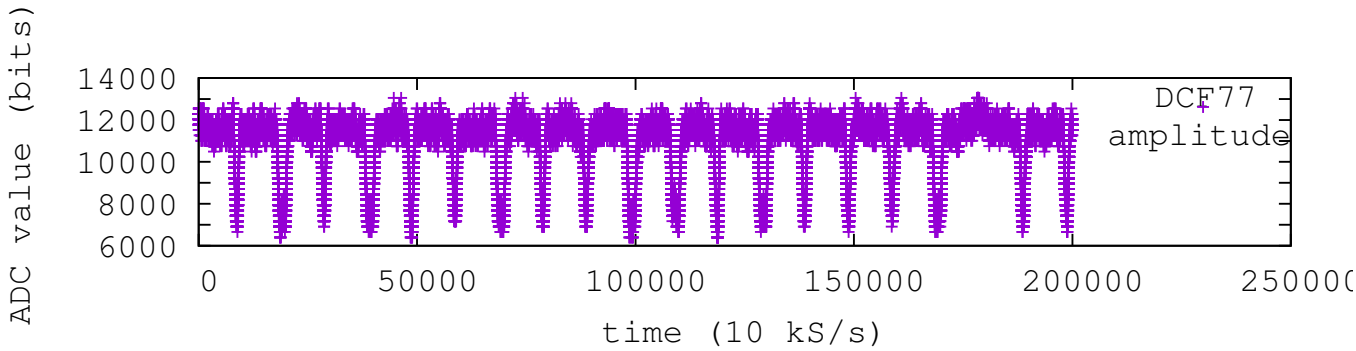
```
# Tue Sep 13 11:15:02 2016

C2:INSP "
  -3840  -3584  -3072  -3328  -3840  -3584  -3840  -3840  -3584  -2816
  -3840  -3584  -3328  -3584  -3584  -3584  -4096  -3584  -3328  -4096
  -3584  -3584  -4096  -3840  -3584  -3584  -3328  -3840  -3584  -3584
  -3584  -4352  -3072  -3584  -4352  -3584  -3840  -4096  -3840  -3840
  -3584  -3584  -3840  -3584  -3840  -3840  -3840  -3840  -4096  -4352
  -3584  -3584  -3584  -3840  -3840  -3840  -3840  -3840  -4096  -3328
...
"
```

2. J.-M Friedt, Cartographier le bout du monde, GNU/Linux Magazine France **185** (Sept. 2015), disponible à `http://jmfriedt.free.fr/lm_cart.pdf`

`gnuplot`[3] exige, pour tracer la courbe, une colonne unique de nombres contenant les valeurs à afficher. Sachant que la commande `zcat` permet d'afficher le contenu d'un fichier compressé tel que disponible à `http://jmfriedt.sequanux.org/1473758102_2.dat.gz`, manipuler le fichier afin de fournir un résultat affichable par `gnuplot`. On pourra (par exemple) pour ce faire

1. **éliminer les 3 premières lignes (étudier le `man` de `tail` pour y arriver) d'entête décrivant la nature des données,**
2. **éliminer la dernière ligne (étudier le `man` de `head` pour y arriver),**
3. **éliminer en bout de chaque ligne le retour chariot (code ASCII 13 – obtenu au clavier en appuyant sur CTRL-V puis CTRL-M), utile sous DOS ou MS-Windows mais inutile sous Unix (constater la présence de ce caractère en visualisant le fichier sous `vim` ou au moyen de `zcat fichier.gz | cat -v | head`),**
4. **mettre bout à bout toutes les lignes résultantes pour en faire une seule longue ligne contenant toutes les valeurs (on noter qu'en utlilisant GNU/Octave ou Matlab, un tel résultat est déjà affichable)**
5. **éliminer les espaces redondants dans la grande ligne ainsi générée (trouver l'option de `tr` qui compresse les répétitions de caractères),**
6. **remplacer les espaces (uniques entre chaque nombre) par un retour chariot pour former une colonne.**



Constater la cohérence des signaux modulés en amplitude émis par DCF77 depuis Mainflingen (près de Francfort) et décrits à
`https://www.ptb.de/cms/en/ptb/fachabteilungen/abt4/fb-44/ag-442/dissemination-of-legal-time/dcf77/dcf77-amplitude-modulation.html`**.**

2.12 Variables

Le shell unix permet de stocker une `valeur` dans une variable `var` par l'assignation `var=valeur`. Le contenu d'une variable s'affiche en préfixant son nom du symbole dollar : `echo $var`

Un certain nombre de variables standard sont utilisées par le shell et doivent être connues pour en expoiter efficacement les fonctionnalités. Parmi les plus utiles, le chemin contenant l'ensemble des répertoires où un exécutable est susceptible de se trouver – `echo $PATH`, ou les répertoires susceptibles de contenir des bibliothèques dynamiques `LD_LIBRARY_PATH`. Le chemin de sa maison est contenu dans `$HOME` tandis que le terminal dans lequel les fenêtres s'affichent est défini dans `$DISPLAY`.

Une variable peut se voir assigner le résultat d'une commande par l'utilisation des *backquotes* : `var=‘commande‘`
Assigner à la variable `a` la valeur de racine de 2, sachant que le calculateur basique `bc` travaille sur des nombres à virgules quand il est appelé avec l'option `-l` et qu'il sait calculer une racine par la fonction `sqrt()`

⚠Attention aux risques de sécurité en préfixant son PATH d'un répertoire personnel : risque d'exécuter un code malicieux dans son arborescence en croyant exécuter une commande système.

Une définition de variable n'est valable *que pour le terminal dans lequel l'affectation est exécutée*, et ne se propage pas aux autres terminaux.

Il est courant de placer les définitions de variables d'environnement dans les fichiers chargés automatiquement à la connexion de l'utilisateur sur un terminal. Ce fichier change de nom selon les shells, mais les plus courants sont `.bashrc` pour le shell bash (sélectionné par défaut) ou `.cshrc` pour le shell csh qui propose une syntaxe à peine plus proche du C. `.profile` n'est lu que lors d'une connexion distante ou la première connexion sur la console, tandis que `.basrc` est lu à la connexion à chaque nouveau terminal.
Consulter le contenu du .bashrc en exécutant `cat $HOME/.bashrc`

3. Le contenu d'un fichier texte `fichier` se trace sous `gnuplot` par `plot 'fichier' title 'legende'` auquel on peut ajouter `set xlabel 'abscisse'` et `set ylabel 'ordonnee'`.

3 Commandes multiples

3.1 Boucle `while`

La boucle `while` (tant que ...) permet notamment d'effectuer des boucles infinies. Un exemple de boucle infinie qui affiche un message toutes les secondes : `while true; do echo "hello";sleep 1;done`

Cette fonction est utile pour tester une opération répétitive depuis le shell, par exemple de faire clignoter une diode.

3.2 Boucle `for`

Itération sur une liste de fichiers : `for i in ‘ls‘; do echo $i;done`

Exemple : changer le format de toutes les photographies dans un répertoire

```
for i in *.jpg; do nom=‘echo $i | sed ’s/jpg/png/g’‘ ;echo $i $nom; convert $i $nom;done
```

Exemple : changer la taille de toutes les photographies dans un répertoire

```
for i in *.jpg; do nom=‘echo $i | sed ’s/\.jpg/petit.jpg/g’‘;echo $i $nom ; convert -scale 50\% $i $nom;done
```

Nous avons vu auparavant (commande `wget`) comment récupérer toutes les images contenues sur le site `http://jmfriedt.sequanux.org/130929_kvad/`. Partant de l'arborescence contenant ces photographies :

1. **réduire toutes les images pour leur donner une largeur de 500 pixels.**
2. **afficher séquentiellement toutes les images contenues dans le répertoire au moyen de la commande `display` (de ImageMagick) en attendant 2 secondes entre chaque image. On notera que la commande `display` accepte pour argument le retard après lequel la prochaine image sera affichée.**

Afficher, au moyen de la commande `xpdf`, tous les fichiers PDF contenus dans un répertoire.

Pour compter, la fonction `seq` permet de générer une séquence de valeurs. La liste des arguments de `for` est ainsi générée comme résultat de l'exécution de la commande `seq` en l'encapsulant entre accents graves (*backquotes*) :

`for i in ‘seq 1 10‘; do echo $i;done`

Créer 10 fichiers, au moyen de la commande `touch`, nommés `toto01` à `toto10`. Comment découvrir l'option pour ajouter le 0 manquant devant les unités?.

3.3 Conditions `if`

Le test se place entre `[[ ]]` : si nous voulons tester l'égalité à 1 d'une variable, nous pouvons utiliser

```
if [[ $a == "1" ]] ; then echo 1; else echo "pas 1";fi
```

Tester cette commande après avoir assigné `a=1` et `a=2`. Est-ce que le résultat observé est cohérent avec nos attentes?