Mélanger les lignes en shell Unix

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 8 février 2011

https://www.bortzmeyer.org/unsort.html

C'est toujours amusant de voir la quantité de logiciels qui existent pour le shell Unix. Hier, j'avais un problème simple, mélanger au hasard les lignes de texte produites par un programme et je craignais qu'il n'existe pas de solution. Au contraire, grâce à des twittériens sympas (et grâce à un moteur de recherche), j'en ai trouvé de nombreuses.

D'abord, le problème : un script shell qui doit se connecter à un serveur de noms parmi ceux faisant autorité pour une zone. La commande dig renvoie une liste de candidats (ici, pour `.fr`) :

```
% dig +short NS fr.
f.ext.nic.fr.
d.nic.fr.
e.ext.nic.fr.
d.ext.nic.fr.
a.nic.fr.
c.nic.fr.
g.ext.nic.fr.
```

et il faut en prendre un au hasard. Minute, dirons certaines personnes qui ont déjà fait du DNS : cela se fait tout seul. BIND mélange automatiquement les réponses (on nomme cela le "*round-robin*"). Certes, mais c'est spécifique à BIND, aucune norme DNS n'exige cela et le résolveur dont je me sers, Unbound, ne le fait pas, et à juste titre <http://www.mail-archive.com/unbound-users@unbound.net/msg00545.html> : c'est le client DNS qui doit mélanger, s'il le souhaite.

Comment le faire en shell ? D'abord, voyons une solution non-portable. La version GNU de `sort` (utilisée par exemple par Debian) a une option `-R` (ou `--random-sort`) qui fait exactement ce qu'on veut (idée de Pascal Bouchareine) :

```
% dig +short NS fr. | sort -R | head -n 1
f.ext.nic.fr.

% dig +short NS fr. | sort -R | head -n 1
e.ext.nic.fr.
```

Autre solution, en Perl (due à Gaël Roualland) :

```
% dig +short NS fr. | perl -MList::Util -e 'print List::Util::shuffle <>' | head -n 1
a.nic.fr.

% dig +short NS fr. | perl -MList::Util -e 'print List::Util::shuffle <>' | head -n 1
c.nic.fr.
```

Avec Ruby (code de Ollivier Robert et Farzad Farid) :

```
% dig +short NS fr. | ruby -e "puts STDIN.readlines.shuffle.first"
g.ext.nic.fr.
```

Et en Python (code de Jérôme Petazzoni) :

```
% dig +short NS fr. | \
  python -c "import random,sys ; print random.choice(sys.stdin.readlines()),"
d.nic.fr.
% dig +short NS fr. | \
  python -c "import random,sys ; print random.choice(sys.stdin.readlines()),"
d.nic.fr.
```

Et puis pourquoi s'arrêter là ? En Scala (code de Eric Jacoboni, non testé) :

```
scala -e 'util.Random.shuffle(io.Source.stdin.getLines) foreach println'
```

On peut souhaiter utiliser plutôt un programme tout fait. Étonnemment, il n'en existe pas moins de **quatre** juste pour cette tâche (en commentaires, j'indique le nom du paquetage Debian contenant ce programme) :

```
# unsort, suggestion de Hauke Lampe
# Paquetage unsort

% dig +short NS fr. | unsort -r | head -n 1
e.ext.nic.fr.

% dig +short NS fr. | unsort -r | head -n 1
d.nic.fr.

# bogosort, suggestion de Hauke Lampe
# Plus de paquetage dans Debian
# <http://www.lysator.liu.se/~qha/bogosort/> ne compile pas tout
# seul, non testé

# shuf, trouvé dans la FAQ Bash (merci à Pierre Chapuis pour l'aide)
# Paquetage coreutils

% dig +short NS fr. | shuf -n 1
g.ext.nic.fr.

% dig +short NS fr. | shuf -n 1
```

```
c.nic.fr.

# randomize-lines, suggestion de Thibaut Chèze
# Paquetage randomize-lines

% dig +short NS fr. | rl -c 1
a.nic.fr.

% dig +short NS fr. | rl -c 1
c.nic.fr.
```

unsort dispose d'options intéressantes, je vous invite à regarder sa page de manuel. Toutes ces commandes nécessitent l'installation d'un programme qui n'est pas forcément présent par défaut. Y a-t-il une solution en shell « pur » ?

Les shells modernes disposent d'une pseudo-variable `RANDOM` qui est plus ou moins aléatoire. On peut donc s'en servir (et c'est la solution que j'ai adopté pour mes scripts) pour tirer au hasard. En m'inspirant d'un excellent cours sur ksh <http://www.dartmouth.edu/~rc/classes/ksh/print_pages.shtml>, j'ai fait :

```
% set $(dig +short NS fr.)
% shift $(($RANDOM % $#))
% echo $1
e.ext.nic.fr.

% set $(dig +short NS fr.)
% shift $(($RANDOM % $#))
% echo $1
d.nic.fr.
```

Ce code affecte le résultat de dig à `$*`, puis décale d'une quantité aléatoire (le `%` est l'opérateur modulo). Il marche sur bash, zsh et ksh. Mais la pseudo-variable `RANDOM` n'est pas normalisée dans Posix (dash, par exemple, ne l'a pas) et le script n'est donc pas vraiment portable. Ollivier Robert me rappelle que FreeBSD a une commande random <http://www.freebsd.org/cgi/man.cgi?query=random&apropos=0&sektion=1&manpath=FreeBSD+8.1-RELEASE+and+Ports&format=html> (dans la section "*games*" donc pas forcément installée) qui peut être utilisée pour générer ce nombre aléatoire, à la place de `$RANDOM`. Si on ne veut dépendre d'aucun programme extérieur, je ne crois pas qu'il existe de solution complètement portable, et qui marche avec n'importe quel shell (par exemple, `/dev/random` n'est sans doute pas normalisé non plus...).

Si on veut du code qui est à peu près sûr de marcher sur tout Unix, il faut se limiter aux outils de base, dont awk fait partie. gbfo me suggère (et ça marche) :

```
% dig +short NS fr. | awk '{t[NR]=$0}END{srand();print t[1+int(rand()*NR)]}'
f.ext.nic.fr.

% dig +short NS fr. | awk '{t[NR]=$0}END{srand();print t[1+int(rand()*NR)]}'
d.ext.nic.fr.
```

Plusieurs autres solutions intéressantes figurent dans l'excellente FAQ de bash <http://mywiki.wooledge.org/BashFAQ/026> (merci à Robin Berjon pour le pointeur). Une des plus amusantes ajoute un nombre aléatoire devant chaque valeur, puis trie, puis retire le nombre ajouté :

```
% randomize() {
        while IFS='' read -r l ; do printf "$RANDOM\t%s\n" "$l"; done |
        sort -n |
        cut -f2-
    }

% dig +short NS fr.  | randomize | head -1
a.nic.fr.

% dig +short NS fr.  | randomize | head -1
f.ext.nic.fr.
```

Attention, ce code ne marche pas avec zsh qui crée un « sous-shell » dans `while`. Comme le dit `ladocumentation`, « *"subshells that reference RANDOM will result in identical pseudo-random values unless the value of RANDOM is referenced or seeded in the parent shell in between subshell invocations"* ». Merci à Manuel Pégourié-Gonnard pour ses explications sur ce point. Il propose un autre code, que je n'ai pas testé :

```
randomize() {
    RANDOM=$RANDOM
    while IFS='' read -r l; do
         printf "$RANDOM\t%s\n" "$l"
    done | sort -n | cut -f2-
}
```