Unix et Programmation Shell
14:00–16:00 W'210

11 janvier 2016

```
 ----------------------------------
/ Vous repondrez aux questions en   \
\ utilisant deux lignes en moyenne. /
 ----------------------------------
       \   ^__^
        \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
```

Q 1. Donner une explication

```
$> foo
bash: foo : commande introuvable
$> PATH=$PATH:bar
$> foo
foobar
```

Il y a un exécutable **foo** dans le répertoire **$HOME/bar** qui affiche "foobar".

Q 2. Deviner le résultat de la commande

```
$> man bash \
    | grep -Eo '\$[A#*0$]{1}'\
      | sort | uniq
```

Il affiche quelques paramètres spéciaux de `bash` soit $#,$*,$0,$$.

Q 3. Que signifie l'acronyme `gawk` version **gnu** d'un célèbre filtre **unix** ? Gnu, Aho, Weinberger, Kernighan.

Q 4. Que doit-on à M. D. McIlroy ? Le pipeline **unix**.

Q 5. Citer 5 types de fichiers présents dans un système de fichier **unix**. Les types rencontrés en cours : régulier, répertoire, blocs, série, tube, lien, et socket.

Q 6. Citer trois shells **unix**. `bash`, `sh`, `csh`, etc...

Q 7. Que décrit la section 1 du manuel ? Les commandes utilisateurs.

Q 8. Donner un exemple de binaire **suid** ? `/bin/passwd`

Q 9. Que fait le script

```
1  #!/bin/bash
2  file =$1 ; shift
3  for n in $* ; do
4      p[$n]=yes
5  done
6  n=1
7  while read line ; do
8      if [ yes = "${p[$n]}" ]; then
9          echo $line
10     fi
11     let n++
12 done < $file
```

Il affiche les lignes de numéros donnés d'un fichier, par exemple, `line.sh foo i j k` affiche les i, j, et k du **foo**. sur la ligne de commande.

Q 10. Ecrire une commande pour trouver les fichiers de suffixe `.h` du répertoire `/usr/include` contenant 3.1415926.

```
grep -rl '3.1415926' --include='*.h'
    /usr/include
```

Q 11. Quel est le résultat de la commande

```
$ for x in {1..100}; do let s+=x;
  done; echo $s;
```

La somme des 100 premiers entiers, soit 5050.

Q 12. Quel service réseau est en relation avec le fichier /etc/resolv.conf ? Domain Name Service.

Q 13. Commenter

```
$ find /bin -name 's*' -type l
/bin/sh
```

`/bin/sh` est le seul lien dont le nom commence par 's', dans le répertoire `/bin`.

Q 14. On considère le script `narco.sh`

```
1 #!/bin/bash
2 for x in $*; do
3     ( sleep $x; echo -n $x ) &
4 done
```

Quel est le résultat de `narco.sh 3 1 4 1 5` ? La commande lance 5 processus en arrière plan. Le second termine en premier et affiche 1, suivent 1, 3, 4 et 5.

Q 15. Citer trois éditeurs de fichiers textes usuels. Par exemple, `vi`, `emacs`, `gedit`, etc...

Q 16. Que fait le script

```
1 for file in *.c ; do
2   if ! gcc $file 2>/dev/null; then
3         break
4   fi
5 done
6 echo $file
```

. Il pointe (imparfaitement) le nom d'une C-source non compilable du répertoire courant.

Q 17. Que fait la commande :

```
$ find ~ -name '*~' | xargs rm -f
```

Elle supprime tous les fichiers de sauvegardes de l'utilisateur.

Q 18. Donner une commande pour supprimer les lignes vides du fichier **foo**.

```
sed -ri '/^$/d' foo
```

Q 19. Donner une commande pour transformer les chaines JJ/MM/AAAA en MM/JJ/AAAA du fichier **bar**, où les lettres J,M,A représentent des chiffres décimaux.

```
sed -ri 's#([0-9]{2})/([0-9]{2})
   /([0-9]{4})#\2/\1/\3# bar
```

Q 20. Expliquer

```
$ mkfifo /tmp/fifo
$ tr 123 456 < /tmp/fifo &
$ echo 1 2 3 4 5 6 > /tmp/fifo
4 5 6 4 5 6
```

La commande (3) écrit 1 2 3 4 5 6 dans le tube créé par (1), le filtre (2) lit dans ce tube et transforme 1 en 4, 2 en 5 et 3 en 6.

```
 -------------------------
/ Ecrire des scripts sans \
\ rature ni surcharge     /
 -------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Q 21. Ecrire un script `wobistdu.sh` pour déterminer sur quel hôte de la grappe **licinfo-1** à **licinfo20** est connecté l'utilisateur toto.

```
1 #!/bin/bash
2 for host in licinfo {1..20}; do
3   if ssh $host "who | grep -q toto" ; then
4     echo $host
5   fi
6 done
```

Q 22. On peut vérifier avec la commande `date` que cette année, le 1er mai sera un dimanche, l'année dernière c'était un vendredi

```
$> date +"%A %D" -d 05/01/2015
vendredi 05/01/15
$> date +"%A %D" -d 05/01/2016
dimanche 05/01/16
```

Ecrire un script `mai.sh` pour compter combien de fois le jour de la fête du travail tombe un jour de week-end sur une période donnée. Par exemple,

```
$> ./mai.sh 1901 2001
28
```

```
1 y=$1
2 while [[ $y != $2 ]]; do
3     date +"%A %m %Y" -d "05/01/$y"
4     let y++
5 done |& grep -cE '(sam|dim)'
```