Unix et Programmation Shell

10 janvier 2015

Vous répondrez aux questions en utilisant deux lignes en moyenne.

Q 1. Commenter

```
$ echo {a..h}{1..8} | wc -w
64
```

Le développement des accolades produit les 64 mots : a1 a2 a3 ... z8.

Q 2. Citer trois paramètres spéciaux correspondant à l'expression régulière : `\$[^1-9]` Par exemple : $0 : nom de commande, $* : liste des arguments, $$: pid, $? : dernier résultat.

Q 3. Citer 5 sections du manuel `unix` : commande, appel système, bibliothèque C, jeux, divers, périphérique et fichiers, format et concention .

Q 4. Citer deux pionniers du système Ken Thompson, Dennis Ritchie. `unix`.

Q 5. Que signifie GUI ? Interface Graphique pour Utilisateur.

Q 6. Citer 5 types de fichiers présents dans un système de fichier. Fichier : standard, répertoire, tube, liens, série, blocs.

Q 7. Décrire deux mécanismes fondamentaux du système UNIX. Le tube quit permet de chainer les processus, la redirection qui permet de manipuler les fichiers d'entrée-sortie.

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

Q 9. Commenter.

```
$ touch x xy xyz
$ find . -name x*
find: les chemins doivent preceder l
   'expression : xy
```

Le développement de * produit la commande `find . -name x xy xyz` qui ne respecte pas l'usage de `find`.

Q 10. Quelle différence entre `fopen` et `open` ? La première est une fonction de la bibliothèque `glibc`, le second un appel système.

Q 11. Commenter

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

Il y a un seul lien dans sous le répertoire `/bin` dont le nom commence par un 's'.

Q 12. Que donne

```
find /etc -perm 000
```

Liste les fichiers fermés pour tout (rwx) et pour tous (ugo).

Q 13. Citer trois commandes externes homonymes d'une commande interne. echo, printf, pwd.

Q 14. Que fait la commande :

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

Elle efface tous les fichiers sous le répertoire courant.

Q 15. Expliquer

```
$ touch ls
$ PATH=.
$ ls
bash: ./ls: Permission non accord e
```

Le shell recherche la commande externe ls dans le répertoire courant, l'exécution échoue car le fichier ls est non exécutable.

Q 16. Expliquer

```
$ unalias x
```

La commande interne supprime l'entrée x de la liste des alias.

Q 17. Ecrire un script `tri.sh` pour trier les arguments de la ligne de commande en mode numérique.

```
1 echo $* | sort -n
```

Q 18. Proposer une explication

```
$ find -name x*
./x.log
$ touch xxx
$ find -name x* 2>/dev/null;
$ echo $?
1
```

Comme plus haut, le développement du nom de fichier * produit une commande find qui échoue.

Q 19. Quel est le résultat de

```
$ echo {1..3} \
       | awk -F2 '{print $2$1}'
```

31

Q 20. Que fait la commande

```
$ sed -irE 's/[ ]+$//' fichier
```

Elle supprime les espaces en fin de ligne dans fichier.

Q 21. Que fait la commande

```
$ grep -vE '^$' src > dst
```

Elle recopie les lignes non vides de src dans dst.

Q 22. Que fait la commande

```
$ grep -rlE '([0-9]+\.){3}[0-9]+' \
  --include=r*.*f /tmp 2>/dev/null
```

Elle recherche les fichiers dont le nom correspond à 'r*.*f' et qui contiennent une chaine de type adresse IPv4.

Q 23. Expliquer

```
$ x=$$; while [ $x != 0 ];do; \
ps --no-headers -p$x -opid,cmd; \
x=$(ps --no-headers -p$x -oppid);\
done

2679 /bin/bash
2678 vim upsh-x-ct-13-1.tex
2571 bash
2049 gnome-terminal
   1 /sbin/init
```

Le script affiche la liste de ces ancêtres.

Q 24. Expliquer

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

Le filtre `tr` en lecture sur le tube remplace les 1 par 4, 2 par 5 et 3 par 6.

Q 25. Le script ci-dessous a été utilisé pour obtenir les traces de la figure (1).

```
#!/bin/bash
strace -ff -e trace=open,read,write,dup2,clone,execve \
   bash -c "$*" |& sed 's/bash.*/bash.../' \
    | grep -vE '(share|lib|stack|^Proc|ELF)' | cut -c1-72 > traces.txt
```

- Combien de processus ont été lancés ? 3
- Quelle commande a été tracée ?

```
cat < /tmp/src | wc -l > /tmp/dst
```

```
execve("/bin/bash...
open("/etc/ld.so.cache", O_RDONLY)      = 3
open("/dev/tty", O_RDWR|O_NONBLOCK|O_LARGEFILE) = 3
open("/proc/meminfo", O_RDONLY)         = 3
read(3, "MemTotal:        1025748 kB\nMemF"..., 1024) = 1024
clone(Process 3605 attached
[pid  3604] clone(Process 3606 attached
[pid  3606] dup2(3, 0)                  = 0
[pid  3606] open("/tmp/dst", O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE, 0666)
[pid  3606] dup2(3, 1)                  = 1
[pid  3606] execve("/usr/bin/wc", ["wc", "-l"], [/* 42 vars */]) = 0
[pid  3605] dup2(4, 1)                  = 1
[pid  3606] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  3605] execve("/bin/cat", ["cat", "/tmp/src"], [/* 42 vars */]) = 0
[pid  3605] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  3606] <... open resumed> )        = 3
[pid  3606] read(3, "# Locale name alias data base.\n#"..., 4096) = 2512
[pid  3606] read(3, "", 4096)           = 0
[pid  3605] open("/tmp/src", O_RDONLY|O_LARGEFILE) = 3
[pid  3605] read(3, <unfinished ...>
[pid  3606] <... open resumed> )        = 3
[pid  3605] <... read resumed> "hello world\n", 32768) = 12
[pid  3605] write(1, "hello world\n", 12) = 12
[pid  3605] read(3, "", 32768)          = 0
[pid  3606] read(0, "hello world\n", 16384) = 12
[pid  3606] read(0, "", 16384)          = 0
[pid  3606] write(1, "1\n", 2)          = 2
--- {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3605, si_status=0, si_
```

FIG. 1 – traces d'une commande.