

Unix et Programmation Shell

10 janvier 2014

Vous répondrez aux questions en utilisant deux lignes en moyenne.

Q 1. Pour les 3 premières sections du manuel : 1 Commandes utilisateur, 2 Appels système, 3 Fonctions de bibliothèque C, citer un exemple de page.

`man` (1), `getpid` (2), `putchar` (3).

Q 2. Quel hacker est à l'origine du projet GNU ?

Richard Stallman.

Q 3. Préciser la terminologie : LINUX.

Nom d'un noyau unix dû à Linus Torwald.

Q 4. Préciser la terminologie : POSIX.

Un des standards pour unix : portable operating system interface (unix).

Q 5. Que signifie : CLI ?

Command Line Interface.

Q 6. Citer un shell populaire.

`sh`, `bash`.

Q 7. Décrire deux mécanismes fondamentaux du système UNIX.

Parmi les fondamentaux présents au niveau du shell : la redirection de fichier et les tubes de communications.

Q 8. Que permet de faire la fonction C `system` ?

Elle permet de lancer une commande au shell par défaut, ce n'est donc pas un appel système !

Q 9. Commenter.

```
$ cat var.sh
cpt=0
( while [ $cpt -lt 3 ] ; do
  echo -n $cpt
  let cpt++
done )
echo $cpt
$ ./var.sh
0120
```

La boucle est lancée dans un sous-shell.

Q 10. Le binaire `run.exe` ouvre le fichier `run.log` en écriture. A priori, qui de `drm`, `ken` ou `pil` peut exécuter `run.exe` sans problème ?

```
-r--rw-r-- 1 ken go run.log
--x--s--x 1 pil go run.exe
drwx----- 1 drm go /home/drm
drwx----- 1 ken yy /home/ken
drwx----- 1 pil xx /home/pil
```

Aucun problème d'exécution pour personne mais `ken` aura des problèmes avec les droits d'écriture. Le bit `suid` sur le groupe permet à `pil` et `drm` d'ouvrir le `run.log`.

Q 11. Proposer une commande plus courte pour réaliser le même travail que :

```
$ find . -name '*' -exec echo {} \;
```

```
$ find
```

Q 12. Commenter la commande

```
$ find / -maxdepth 1 -perm +1000
-printf "%p-%m-%M\n"
/tmp-1777-drwxrwxrwt
```

Recherche à la racine sans descente récursive des fichiers possédant un bit spécial. Typiquement le fichier `/tmp` avec son bit collant.

Q 13. Préciser au moins trois variables d'environnement initialisées par `login`.

SHELL, PATH, HOME.

Q 14. Que fait la commande :

```
$ find -name '*.html' -exec \
sed -i 's/USTV/UTLN/g' '{}' \;
```

Pour chaque fichier `.html` de l'arborescence courante, toutes les occurrences de `USTV` sont remplacées par `UTLN`.

Q 15. Expliquer

```
$ ls $( find . -name 'ls' )
./ls
```

Nous sommes probablement dans le répertoire `/bin`.

Q 16. Expliquer

```
$ unalias x
$ x
bash:x:commande introuvable
$ source x
$ x
hello
```

Le fichier `x` du répertoire courant contient la ligne :

```
alias x='echo hello '
```

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

```
#!/bin/bash
echo $* | tr ' ' '\n' | sort -n
```

Q 18. Proposer une explication

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

Le développement de `x*` provoque une erreur du `find`.

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
```

supprime les espaces en fin de ligne de fichier.

Q 21. Que fait la commande

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 ?
- Quelle commande a été tracée ?

2 par la commande tracée :

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

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

copie les lignes non vides de `src` dans `dst`.

Q 22. Que fait la commande

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

recherche dans l'arborescence `tmp` les fichiers dont le nom concorde avec le motif `r*.f` contenant une 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
```

La commande `ps` est utilisée pour déterminer de proche en proche l'ascendance du processus courant.

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 de remplacement de caractères `tr` est appliqué à la sortie du tube `/tmp/fifo`. Le tout est équivalent à

```
$ echo 1 2 3 4 5 6 | tr 123 456
4 5 6 4 5 6
```

```

1  execve("/bin/bash...
2  open("/etc/ld.so.cache", O_RDONLY) = 3
3  open("/dev/tty", ORDWR|O_NONBLOCK|O_LARGEFILE) = 3
4  open("/proc/meminfo", O_RDONLY) = 3
5  read(3, "MemTotal:      1025748 kB\nMemF"... , 1024) = 1024
6  clone(Process 3605 attached
7  [pid 3604] clone(Process 3606 attached
8  [pid 3606] dup2(3, 0) = 0
9  [pid 3606] open("/tmp/dst", O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE, 0666)
10 [pid 3606] dup2(3, 1) = 1
11 [pid 3606] execve("/usr/bin/wc", ["wc", "-l"], [/* 42 vars */]) = 0
12 [pid 3605] dup2(4, 1) = 1
13 [pid 3606] open("/etc/ld.so.cache", O_RDONLY) = 3
14 [pid 3605] execve("/bin/cat", ["cat", "/tmp/src"], [/* 42 vars */]) = 0
15 [pid 3605] open("/etc/ld.so.cache", O_RDONLY) = 3
16 [pid 3606] <... open resumed> ) = 3
17 [pid 3606] read(3, "# Locale name alias data base.\n#"..., 4096) = 2512
18 [pid 3606] read(3, "", 4096) = 0
19 [pid 3605] open("/tmp/src", O_RDONLY|O_LARGEFILE) = 3
20 [pid 3605] read(3, <unfinished ...>
21 [pid 3606] <... open resumed> ) = 3
22 [pid 3605] <... read resumed> "hello world\n", 32768) = 12
23 [pid 3605] write(1, "hello world\n", 12) = 12
24 [pid 3605] read(3, "", 32768) = 0
25 [pid 3606] read(0, "hello world\n", 16384) = 12
26 [pid 3606] read(0, "", 16384) = 0
27 [pid 3606] write(1, "1\n", 2) = 2
28 --- {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=3605, si_status=0, si_

```

FIG. 1 – traces d’une commande.