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.

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

Q 3. Préciser la terminologie : LINUX.

Q 4. Préciser la terminologie : POSIX.

Q 5. Que signifie : CLI ?

Q 6. Citer un shell populaire.

Q 7. Décrire deux mécanismes fondamentaux du système UNIX.

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

Q 9. Commenter.

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

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
```

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

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

Q 12. Commenter la commande

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

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

Q 14. Que fait la commande :

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

Q 15. Expliquer

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

Q 16. Expliquer

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

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

Q 18. Proposer une explication

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

Q 19. Quel est le résultat de

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

```
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.

Q 20. Que fait la commande

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

Q 21. Que fait la commande

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

Q 22. Que fait la commande

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

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
```

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
```

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

```
#!/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 ?