Unix et Programmation Shell

22 juin 2015

Vous répondrez aux questions en utilisant deux lignes en moyenne.

Q 1. Donner une commande pour compter les lignes vides du fichier **foo** ?

Q 2. Que décrit la section 1 du manuel **unix**.

Q 3. Le chercheur McIlroy a inventé un concept fondamental du système **unix**. Lequel ?

Q 4. Que signifie l'acronyme CLI ?

Q 5. Commenter.

```
$> PS1="$PS1$PS1$PS1$PS1"
$> $> $> $>
```

Q 6. Quel est le résultat de la commande

```
$> echo {0,1}{0,1}{0,1}
```

Q 7. Donner un exemple d'appel système impliqué dans une redirection.

Q 8. Donner un exemple d'appel système impliqué dans un pipeline.

Q 9. Quelle est la particularité du binaire `/bin/passwd` ?

Q 10. La commande qui suit affiche une liste de fichiers dont `/etc/resolv.conf`. Pourquoi ?

```
$> grep -lE '([0-9]{1,3}\.)
   {3}[0-9]{1,3}' /etc/*
```

Q 11. Quelle différence entre `fclose` et `close` ?

Q 12.

```
$> stat --format='%n %A' /tmp/foo
/tmp/foo dr-xr-xr-x
$> rm /tmp/foo/bar
rm: impossible de supprimer
/tmp/foo/bar:
Permission non accordee
```

Comment effacer le fichier ?

Q 13. Deviner le résultat de la commande :

```
$> find /bin -executable -name 'p?'
```

Q 14. Quelle est l'utilité de la commande `wget` ?

Q 15. Que fait le script

```
1 #!/bin/bash
2 sum=0
3 while [ $# -gt 0 ] ; do
4     let sum+=$1
5     shift
6 done
7 echo $sum
```

Q 16. Que fait la commande :

```
$> find -name '*' -exec ls {} \;
```

Q 17. Quelle est l'utilité de la variable `PATH` ?

Q 18. Deviner le résultat de la commande

```
$> ls / | head -5
```

Q 19. La commande `ssh` permet d'ouvrir un shell sur un hôte distant. Que signifie l'acronyme SSH ?

Q 20. Expliquer

```
$> echo --help
--help
$> enable -n echo
$> echo --help
Utilisation : echo [SHORT-OPTION]...
    [STRING]...
...
```

Q 21. Quel est le résultat de

```
$> echo {0..9}    \
          | awk   '{print $3$1$2$6}'
```

Q 22. Que fait la commande

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

```
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:         508520 kB\nMemF"..., 1024) = 1024
clone(Process 2325 attached
[pid  2324] clone(Process 2326 attached
[pid  2324] clone(Process 2327 attached
[pid  2327] dup2(4, 0)                  = 0
[pid  2326] dup2(3, 0)                  = 0
[pid  2326] dup2(5, 1)                  = 1
[pid  2327] open("/tmp/count", O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE, 066
[pid  2327] dup2(3, 1)                  = 1
[pid  2327] execve("/usr/bin/wc", ["wc", "-c"], [/* 52 vars */]) = 0
[pid  2325] dup2(4, 1)                  = 1
[pid  2326] execve("/bin/cat", ["cat"], [/* 52 vars */]) = 0
[pid  2325] execve("/bin/echo", ["/bin/echo", "hello", "world"], [/* 52
[pid  2327] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  2326] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  2325] open("/etc/ld.so.cache", O_RDONLY) = 3
[pid  2325] <... open resumed> )        = 3
[pid  2326] read(0,  <unfinished ...>
[pid  2327] <... open resumed> )        = 3
[pid  2327] read(3, "# Locale name alias data base.\n#"..., 4096) = 2512
[pid  2325] write(1, "hello world\n", 12 <unfinished ...>
[pid  2327] read(3,  <unfinished ...>
[pid  2325] <... write resumed> )       = 12
[pid  2326] <... read resumed> "hello world\n", 32768) = 12
[pid  2326] write(1, "hello world\n", 12) = 12
[pid  2326] read(0, "", 32768)          = 0
[pid  2327] <... read resumed> "", 4096) = 0
[pid  2327] read(0, "hello world\n", 16384) = 12
[pid  2327] read(0, "", 16384)          = 0
[pid  2327] write(1, "12\n", 3)         = 3
--- {si_signo=SIGCHLD, si_code=CLD_EXITED, si_pid=2326, si_status=0, si_
```

FIG. 1 – traces d'une commande.

Q 23. Que fait la commande

```
$> grep -Eo '[0-9]+' src
```

Q 24. Donner le temps d'exécution de la commande

```
$> sleep 10 &  sleep 20
```

et celui de la commande

```
$> sleep 10 && sleep 20
```

Q 25. Ecrire un script pour calculer $n!$.

Q 26. 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 ?