Unix et Programmation Shell

7 janvier 2013

Vous répondrez aux questions en utilisant deux lignes en moyenne.

Q 1. Préciser l'origine de UNIX ?

Le système UNIX a été créé au début des années 70 par Ken Thompson à partir du système MULTICS du Bell Lab.

Q 2. Qu'est-ce que GNU/linux ?

Un système d'exploitation fondé sur les utilitaires GNU et le noyau Linux. En effet, le noyau GNU n'est pas encore opérationnel.

Q 3. Citer 3 UNIX populaires ?

Dans l'ordre d'apparition : System V, BSD, GNU/Linux, OSX, Android.

Q 4. Qu'est que Ubuntu ?

Une distribution GNU/linux.

Q 5. Que signifie : CLI ?

Command Line Interface.

Q 6. Qu'est-ce qu'un shell ?

Un interprète de commande qui d'accéder aux fonctionnalités d'un système. Parmi les plus populaires : sh, bash.

Q 7. Décrire deux mécanismes fondamentaux du système UNIX.

La redirection de fichier et les tubes de communication.

Q 8. Préciser les notions de commande interne et externe

Du point de vue du SHELL : une commande interne est une fonction de l'interprète. Une commande externe est un exécutable PATH-accessible.

Q 9. Préciser 5 variables d'environnement classiques.

vus en TPs : USER, HOME, PATH, PWD, SHELL, HISTFILE, HISTSIZE. A défaut

```
$ env | sed 's/=.*//' | head -5
```

Q 10. Résultats des commandes

```
echo $( 1 + 1 )
echo $(( 1 + 1 ))
```

La première première provoque une erreur, la seconde affiche 2.

Q 11. Expliquer en détail

```
$ oops
bash:oops: commande introuvable
```

`bash` n'a pas trouvé `oops` dans les alias, les comandes internes, et les commandes externes.

Q 12. Proposer une explication

```
$ cat $( cat  cat )
cat
cat
cat
cat
```

Plusieurs solutions, par exemple, un fichier `cat` dans le répertoire courant ayant pour contenu `cat cat`.

Q 13. Que fait la commande

```
$ find ~guest -name "*.pdf" \
  -exec lpr {} \;
```

Lance l'impression de tous les fichiers `pdf` présents dans le répertoire de l'utilisateur `guest`.

Q 14. Que fait la commande

```
$ sed -i 's/[0-9]*/(NB)/g' foo
```

Toutes les chaînes numériques du fichier `foo` sont remplacées par la chaîne (NB).

Q 15. Que fait la commande

```
$ sed -E 's/[0-9]{2}$/foo/' bar
```

Les chaines de 2 chiffres en fin de ligne du fichier `bar` sont remplacées par "foo".

Q 16. Expliquer

```
$ grep -ch bizarre .b*h*h*
0
$ history -w
$ grep -ch bizarre .b*h*h*
1
```

Visiblement, le nom du fichier historique des commandes correspond à l'expression régulière. La chaine "bizarre" apparaît forcément dans l'historique aprés l'exécution de la première commande.

Q 17. Résultat de la commande

```
$ echo {1..5} | wc -c
```

Comme "1..5" est développé en 1 2 3 4 5, tenant compte des espaces et NL, `wc -c` donne 10.

Q 18. Que fait la commande

```
$ grep -r -E '[a-z]+' \
          --include=*.tex /tmp
```

Affiche les lignes des fichiers `.tex` de l'arborescence `/tmp` qui contiennent une chaine alphabétique.

Q 19. Expliquer

```
$ ps -eoppid,pid,cmd | grep $$
26121 12505 bash
12505 12884 ps -eoppid,pid,cmd
12505 12885 grep 12505
```

Les processus ps et grep sont fils du même processus bash, ils ont le même PPID : 12505 dans l'exemple qui est le PID de la commande composée : contenu de la variable spéciale $$.

Q 20. Expliquer

```
$ mkfifo /tmp/fifo
$ tr world hello < /tmp/fifo
$ echo hello world > /tmp/fifo
helle hello
```

La commande `tr` effectue des remplacements caractère par caractère : w devient h, o↦e, r↦l, l↦l, o↦d sur la sortie du tube `/tmp/fifo`. Les arguments de `echo` sont envoyés dans le tube : le mot "hello" est transformé en "helle" et "world" est inchangé.

```
flags=read,write,open,dup2,execve,clone
strace -fe trace=$flags bash -c "$1" &> traces.all
grep -vE '(lib|share|ELF|\.so|-1)' traces.all \
   | sed 's/bash.*/;-)/' | cut -c1-75 > traces.txt
```

Q 21. Le script ci-dessus a été utilisé pour obtenir la trace ci-dessous.

- Quelle commande a probablement été tracée ?
  C'est la commande `cat </tmp/src >/tmp/dst`.
- Commenter les lignes 1–5.
  Chargement des bibliothèques dynamiques.
- Commenter les lignes 7–15.
  Recherche des commandes externes `bash` puis `cat`.
- Commenter les lignes 16–17.
  Lancement d'un processus fils.

```
1  open("/etc/ld.so.cache", O_RDONLY)      = 3
2  close(3)                                = 0
3  open("/lib64/libtinfo.so.5", O_RDONLY)  = 3
4  read(3, "\177ELF\2\1\1\0\0\0"....., 832) = 832
5  close(3)                                = 0
6  ...
7  stat("/usr/local/bin/bash", 0x7fff9376c830) = -1
8      (No such file or directory)
9  stat("/usr/bin/bash", 0x7fff9376c830)   = -1
10     (No such file or directory)
11 stat("/bin/bash", {st_mode=S_IFREG|0755, st_size=927096, ...}) = 0
12 ...
13 stat("/usr/bin/cat", 0x7fff9376c7f0)    = -1
14     (No such file or directory)
15 stat("/bin/cat", {st_mode=S_IFREG|0755, st_size=51328, ...}) = 0
16 Process 13510 attached
17 Process 13509 suspended
18 [pid 13510] open("/tmp/dst", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
19 [pid 13510] dup2(3, 1)                   = 1
20 [pid 13510] close(3)                    = 0
21 [pid 13510] open("/etc/ld.so.cache", O_RDONLY) = 3
22 [pid 13510] close(3)                    = 0
23 [pid 13510] open("/lib64/libc.so.6", O_RDONLY) = 3
24 [pid 13510] read(3, "\177ELF\2     "..., 832) = 832
25 [pid 13510] close(3)                    = 0
26 [pid 13510] open("/tmp/src", O_RDONLY)  = 3
27 [pid 13510] read(3, "hello\n", 32768)   = 6
28 [pid 13510] write(1, "hello\n", 6)      = 6
29 [pid 13510] read(3, "", 32768)          = 0
30 [pid 13510] close(3)                    = 0
31 [pid 13510] close(1)                    = 0
32 [pid 13510] close(2)                    = 0
33 Process 13509 resumed
34 Process 13510 detached
35 --- SIGCHLD (Child exited) @ 0 (0) ---
```