Unix et Programmation Shell
corrigé

10 juin 2013

Vous répondrez aux questions en par deux lignes en moyenne.

Q 1. Citer deux informaticiens à l'origine de `GNU/linux`.

R. Stallmann (GNU), L. Torwald (linux).

Q 2. Préciser trois acronymes du monde `unix`.

awk, Bash, pid, POSIX, sed...

Q 3. La commande `uname` permet d'obtenir des informations sur le système. Lesquelles ?

Identification de l'hôte : nom, noyau, système d'exploitation, architecture, distribution...

Q 4. Donner un exemple de système d'exploitation récent basé sur un noyau `linux`.

android.

Q 5. Rappeler l'organisation générale du système de fichiers `unix`.

Sous la racine : `boot`,`dev`,`proc`, `root`, `etc` (configuration), `sbin`,`home` (utilisateur),`lib`, `share`, `bin` (exécutable), `tmp`, `usr`.

Q 6. Quelle est la particularité du répertoire `/proc` ?

Il s'agit d'un répertoire en mémoire vive utilisé pour stocker données des processus notamment.

Q 7. Décrire avec précision le rôle d'une ligne de `shell` commençant par le caractère #.

En général, il s'agit d'un commentaire. En première, ligne, il introduit le `shebang`.

Q 8. Préciser 7 commandes internes populaires du shell `bash`.

`alias`, `echo`, `cd`, `help`, `history`, `pwd`, `time`.

Q 9. Détailler

```
$ ls -il /tmp/fb.sh
48744 -rwxr-x---x 1 foo bar 35
          6 juin  10:23 /tmp/fb.sh
```

Le fichier `/tmp/fb.sh` d'inode 48744 (35 octets) appartient à foo et au groupe bar. Le propriétaire à tous les droits sur ce fichier qui est exécutable par tout le monde, et ouvert en lecture pour le groupe.

Q 10. Commenter avec précision le résultat de la commande ci-dessous :

```
$/usr/bin/time --format="cpu=%U"\
     time -p sleep 2
real 2.00
user 0.00
sys 0.00
cpu=0.00
```

`time` existe au niveau interne et externe. Dans l'exemple, les premières information de temps sont données par la commande interne.

Q 11. Proposer une explication

```
$ alias xp='if ! cat $((x++))
    2>/dev/null;then echo -n $x;fi'
$ xp;xp;xp;xp
12trois
4
```

Une variable x initialisée à 0, un fichier nommé "2" contenant le mot "trois".

Q 12. Que fait la commande

```
grep -rE '([0-9]{1,3}\.){3}[0-9]+'
 --include=r*.*f /etc 2>/dev/null
```

Dans la plupart des cas, cette commande écrira au moins une ligne. Pourquoi ?

Recherche récursive dans `/etc` sur les fchiers correspondants à "r*.*f" des lignes contenant une adresse IPv4. En particulier, la ligne `nameserver` du fichier `resolv.conf`

Q 13. Pour supprimer correctement les balises `html` du fichier `index.html`, quelle est la bonne commande ?

```
$ sed 's/<[^>]*>//' index.html
```

```
$ sed 's/<.*>//' index.html
```

```
$ sed 's/<[^>]*>//g' index.html
```

```
$ sed 's/<.*>//g' index.html
```

La troisième.

Q 14. Commenter

```
[/tmp/ici]$
echo 'cd $1;pwd'>cd;chmod u+x cd;
[/tmp/ici]$ mkdir /tmp/labas
[/tmp/ici]$ ./cd  /tmp/labas
/tmp/labas
[/tmp/ici]$
```

Q 15. Dans l'exemple ci-dessous, sur quel hôte est exécuté la commande `who` ?

```
[]$ ssh  pl@192.168.0.100  who
pl@192.168.0.100's password: ***
pl   tty1    2013-06-06 07:57
pl   pts/1   2013-06-06 13:19
pl   pts/2   2013-06-06 13:05
pat  pts/3   2013-06-06 13:32
```

Rappeler les étapes à suivre pour exécuter des commandes sur une machine distantes sans fournir de mot de passe.

pl exécute `who` sur 192.168.0.100. Il convient de créer une paire de clés privée/publique (`sshkeygen`), et de déposer la clé publique sur les hôtes ciblés (`ssh-copy-id`).

Q 16. Ecrire un script : `whereis` *user hosts*, pour déterminer sur quel hôte est connecté un utilisateur donné.

- *user* : utilisateur
- *hosts* : fichier contenant la liste des hotes accessibles par `ssh`.

Utiliser les commandes : `ping`, `ssh` et `who`.

```
#!/bin/bash
while read host
do
    if ping -qc1 $host
        then
        if [ ssh $host "who  |  grep  $1" ]
            then
                echo $1 on $host
            else
                echo $host
        fi
        else
        echo $2 absent
    fi
done < $2
```

Q 17. Le script (**??**) a été utilisé pour obtenir les traces (**??**).

- Commenter ce script.
  Le script `strace` les appels système du type `flags` d'une commande passer via `bash`. La sortie est filtrée par une combinaison `grep`/`sed`.
- Quelle commande de débogage a été utilisée ?
  `strace`
- Donner un autre exemple de commande de débogage.
  `ltrace`
- Quelle commande a été tracée ?
  `cat </tmp/src | wc -l >/tmp/dst`
- Représenter shématiquement les processus, fichiers et descripteurs de fichiers mis en jeux par la commande tracée.

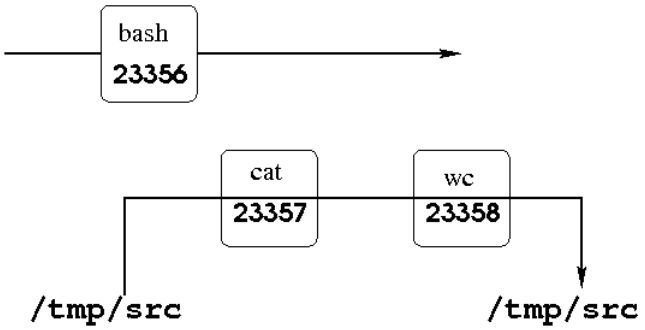

FIG. 1 – pid, descripteurs.

```
1  execve("/bin/;-)
2  open("/dev/tty", O_RDWR|O_NONBLOCK)      = 3
3  open("/proc/meminfo", O_RDONLY)          = 3
4  read(3, "MemTotal:        1798268 kB\nMemF"..., 1024) = 1024
5  clone(Process 23357 attached
6  child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child
7  [pid 23356] clone(Process 23358 attached
8  child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child
9  [pid 23357] dup2(4, 1)                   = 1
10 [pid 23358] dup2(3, 0)                   = 0
11 Process 23356 suspended
12 [pid 23357] open("/tmp/src", O_RDONLY)   = 3
13 [pid 23357] dup2(3, 0)                   = 0
14 [pid 23357] execve("/bin/cat", ["cat"], [/* 53 vars */]) = 0
15 [pid 23358] open("/tmp/dst", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
16 [pid 23358] dup2(3, 1)                   = 1
17 [pid 23358] execve("/usr/bin/wc", ["wc", "-l"], [/* 53 vars */] <unfinished
18 [pid 23358] <... execve resumed> )       = 0
19 [pid 23357] read(0, "gnu posix Bash sed awk\n", 32768) = 23
20 [pid 23357] write(1, "gnu posix Bash sed awk\n", 23 <unfinished ...>
21 [pid 23357] <... write resumed> )        = 23
22 [pid 23358] <... open resumed> )         = 3
23 [pid 23357] read(0, "", 32768)           = 0
24 Process 23356 resumed
25 Process 23357 detached
26 [pid 23358] read(3, "# Locale name alias data base.\n#"..., 4096) = 2512
27 Process 23356 suspended
28 [pid 23358] read(3, "", 4096)            = 0
29 [pid 23358] read(0, "gnu posix Bash sed awk\n", 16384) = 23
30 [pid 23358] read(0, "", 16384)           = 0
31 [pid 23358] write(1, "1\n", 2)          = 2
32 Process 23356 resumed
33 Process 23358 detached
34 --- SIGCHLD (Child exited) @ 0 (0) ---
```

FIG. 2 – capture des appels systèmes.

```
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
```

FIG. 3 – script de traçage.