FICHE 13 - Processus

```
find / -type f -exec grep -Iq . {} \; -and -exec cat {} \;
```

Comment interpréter cette commande ?

`find` → recherche tous les fichiers dans toute l'arborescence

`find options path -exec cmd1 \; -and -exec cmd2 \;` → exécute la commande cmd1 sur les résultats du find ; -and = ET logique : cmd2 n'est pas évaluée si cmd1 est fausse (i.e. renvoie un exit code ≠ 0).

`grep .` → matche les lignes contenant n'importe quel caractère (donc sans les lignes vides)

`grep -I` → ne pas traiter les fichiers binaires

`grep -q` → ne pas afficher les résultats normaux (*quiet*)

`cat` → concatène les lignes trouvées (pour lesquelles ni *find* ni *grep* n'ont renvoyé une erreur) pour reconstituer le contenu des fichiers

Conclusion : cette commande affiche le contenu (sauf les lignes vides) de tous les fichiers réguliers non binaires du système de fichiers.

1. ***commande &***
   a. **jobs**
   b. on ne voit rien car les résultats de *find* sont affichés dans le terminal, cela va tellement vite qu'on ne voit pas ce qu'on tape. Jobs est pourtant bel et bien lancé, mais son affichage se mélange avec celui du find.
   c. *fg* puis *Ctrl-C* pour arrêter l'exécution
   d. ***commande > find.txt 2> /dev/null &***
      jobs
2. **sleep 600 &** [1]
   a. **ps -l** (la deuxième colonne est l'état : « S » pour *state*)

      Process State Codes :
      R running or runnable (dans file d'exécution)
      D non interruptible sleep (normalement entrées/sorties)
      S interruptible sleep (en attente d'un événement)
      T stopped (par un signal)
      Z zombie (<defunct>)

   b. S → le processus dort (c'est la commande *sleep* !)
   c. **fg** [%1 optionnel]

[1] Remarque: *top* ne peut être envoyé en background car cette commande est faite pour interagir avec l'utilisateur, elle s'arrête donc quand elle est envoyée en bg (cf. https://unix.stackexchange.com/questions/396634/top-top-top-commands-in-linux)

d. **Ctrl-Z** → affiche : [1]+ Stopped sleep 600

ps -l → T pour *stopped*

e. **bg** [%1 optionnel] → affiche : [1]+ sleep 600 &

f. **jobs** → affiche : [1]+ Running sleep 600 &

g.
```
sleep 600 &
sleep 600 &
sleep 600 &
sleep 600 &
jobs
```

h.
```
fg 2
CTRL+C
jobs
```

i.
```
fg
CTRL+C
```
} 3 fois

3. **chmod +x timer ; ./timer**

a. CTRL+C => ignoré

b. CTRL+Z => ignore

c. Il faut lancer un autre shell

identifier le PID du processus :

ps -u (-u : les processus de l'utilisateur courant)

OU **ps -ef | grep timer** (-e : tous les processus ; -f : format complet)

kill -9 PID

4. **ps -l** → y trouver le PID (et le PPID) de bash

ps -l *bash_PPID* OU **pstree -ps *bash_PID*** OU **pstree -p $USER**

(-p : show PIDs ; -s : show parent processes of specified process ; $USER : voir tous mes processus)

Si vous êtes connecté en ssh sur une machine Linux, le processus parent du bash est **sshd** = le démon qui reçoit la connection SSH (de la commande ssh ou de putty)

[dans terminal Ubuntu : gnome-terminal]

5.

a. **ps -ef** (premier exemple présenté dans la page de manuel de ps)

b. **ps -ef | grep systemd** [2]

[2] Remarquez que, si on recherchait le mot exact « systemd », nous pourrions utiliser le word boundary \b : `grep "\bsystemd\b"`. Mais dans ce cas, la commande `grep` elle-même disparaît des résultats puisque le mot « systemd » n'y apparaît pas exactement :
`anthony+ 32243 23931  4 13:00 pts/0  00:00:01 grep --color=auto \bsystemd\b`

c. `ps -ef | grep systemd | grep -v grep`

   (grep -v: inverser la mise en correspondance)

d. `ps -ef | grep systemd | grep -v grep | tr -s ' '`

   (tr -s : remplacer chaque séquence par une seule occurrence)

e. `ps -ef | grep systemd | grep -v grep | tr -s ' ' | cut -d " " -f 8`