1) Env et printenv renvoient les variables d'environnement existantes. La seule différence est la dernière ligne qui se termine soit par env soit par  printenv selon celle utilisée

2)

```
env > env.txt
```

3) PATH est la variable contenant tous les outils permettant d'exécuter un programme séparés par des semi colon
SHELL contient le nom du shell utilise
HOME contient le chemin d'accès complet du répertoire personnel
USER contient l'id numérique de l'utilisateur
LOGNAME contient le nom de la personne connectée

4) On va créer les variables en les assignant avec « = » puis on les appelle avec « $ »

5) On va utiliser « export » pour mettre les variables en tant que variables d'environnement

```
22113572t@ST-2019003264:~$ NOM=voisin
22113572t@ST-2019003264:~$ PRENOM=michel
22113572t@ST-2019003264:~$ AGE=40
22113572t@ST-2019003264:~$ PRENOM=michel $AGE
22113572t@ST-2019003264:~$ NOM=voisin
22113572t@ST-2019003264:~$ export PRENOM
22113572t@ST-2019003264:~$ export NOM
22113572t@ST-2019003264:~$ bash
22113572t@ST-2019003264:~$ echo $NOM
voisin
22113572t@ST-2019003264:~$ echo $PRENOM
michel
22113572t@ST-2019003264:~$ echo $AGE
```

6)

```
22113572t@ST-2019003264:~$ NOM=Mael
```

7)  La valeur de NOM n\a pas change dans le shell parent car la variable a été modifiée dans un autre environnement

8) On enlève a une variable son statut d'environnement en rajoutant n comme option a « export »

```
22113572t@ST-2019003264:~$ export -n PRENOM
22113572t@ST-2019003264:~$ bash
22113572t@ST-2019003264:~$ echo PRENOM
PRENOM
22113572t@ST-2019003264:~$ echo $PRENOM
```

9)

```
22113572t@ST-2019003264:~$ unset AGE
22113572t@ST-2019003264:~$ echo $AGE
```

10) La variable PATH est construite en sorte que les paths les plus importants soient mis au début de la liste. Pour que l'extraction soit plus rapide car c'est les plus importants qui seront en général demandes

11) Pour exécuter directement le fichier, on utilise la syntaxe « ./

12) On rajoute le path « . » avant le reste de PATH sépare par un semi

```
22113572t@ST-2019003264:~$ mkdir Tp2
22113572t@ST-2019003264:~$ echo echo bonjour $NOM > ls
22113572t@ST-2019003264:~$ cat ls
echo bonjour voisin
22113572t@ST-2019003264:~$ chmod a+x ls
22113572t@ST-2019003264:~$ ./ls
bonjour voisin
```

```
22113572t@ST-2019003264:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
22113572t@ST-2019003264:~$ PATH=.:$PATH
22113572t@ST-2019003264:~$ echo $PATH
.:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

colon

13) ls exécute désormais notre programme en priorité

15) ps -aef montre tous les processus en cours d'exécution. La différence entre l'option aef et l'argument aux est simplement syntaxique dans le texte renvoyé

16) Le propriétaire est indique dans la colonne USER, le numéro(identifiant) est dans la colonne PID et le statut est indique dans la colonne STAT

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 169384 13164 ?        Ss   13:32   0:02 /sbin/init sp
root           2  0.0  0.0      0     0 ?        S    13:32   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   13:32   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   13:32   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   13:32   0:00 [slub_flushwq
root           6  0.0  0.0      0     0 ?        I<   13:32   0:00 [netns]
```

17) Les processus lance par l'utilisateur sont affiches par la commande « ps »

```
22113572t@ST-2019003264:~$ ps
    PID TTY          TIME CMD
   2776 pts/1    00:00:00 bash
  23755 pts/1    00:00:00 ps
```

18) La commande top montre par défaut en temps réel la consommation des processus en cours. On peut modifier l'affichage en utilisant la fonction désignée pour avec « f »
UID indique quel utilisateur utilise le processus
VIRT mémoire virtuelle utilisée

```
  PID   UID UTIL.      GID    VIRT    RES    SHR S  %CPU  %MEM     TEMPS+ COM.
28751 30686 2211357+   100 2594256 231488 107224 S   2,3   1,4   0:28.14 Iso+
  981     0 root         0 2201056 146380  85192 S   1,7   0,9   1:37.91 Xorg
 1835 30686 2211357+   100 4254200 488768 213572 S   0,7   3,0   2:12.09 x-w+
34490 30686 2211357+   100  550456  50316  39864 S   0,7   0,3   0:00.25 xfc+
   58     0 root         0       0      0      0 S   0,3   0,0   0:02.63 kso+
 1482 30686 2211357+   100 1404532 111400  84140 S   0,3   0,7   0:18.00 xfw+
10234     0 root         0       0      0      0 I   0,3   0,0   0:00.37 kwo+
33793 30686 2211357+   100   22168   6272   4096 R   0,3   0,0   0:00.36 top
    1     0 root         0  169384  13164   8940 S   0,0   0,1   0:02.95 sys+
    2     0 root         0       0      0      0 S   0,0   0,0   0:00.00 kth+
    3     0 root         0       0      0      0 I   0,0   0,0   0:00.00 rcu+
    4     0 root         0       0      0      0 I   0,0   0,0   0:00.00 rcu+
    5     0 root         0       0      0      0 I   0,0   0,0   0:00.00 slu+
    6     0 root         0       0      0      0 I   0,0   0,0   0:00.00 net+
    8     0 root         0       0      0      0 I   0,0   0,0   0:00.00 kwo+
   10     0 root         0       0      0      0 I   0,0   0,0   0:00.00 mm_+
   11     0 root         0       0      0      0 I   0,0   0,0   0:00.00 rcu+
```

19)

```
  PID UTIL.      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TEMPS+ COM.
    1 root       20   0  169520  13248   9024 S   0,0   0,1   0:00.98 systemd
```

Le processus systemd est le premier processus lance par Linux, c'est de ce processus dont les sont issus ceux lances par l'utilisateur.

20)

```
22113572t@ST-2019003175:~/Tp2$ ./testboucle.sh
Tour de boucle n° 1
Attente d'une seconde…
Tour de boucle n° 2
Attente d'une seconde…
Tour de boucle n° 3
Attente d'une seconde…
Tour de boucle n° 4
Attente d'une seconde…
Tour de boucle n° 5
Attente d'une seconde…
Tour de boucle n° 6
Attente d'une seconde…
Tour de boucle n° 7
Attente d'une seconde…
Tour de boucle n° 8
Attente d'une seconde…
Tour de boucle n° 9
Attente d'une seconde…
^C22113572t@ST-2019003175:~/Tp2$
```

Le programme continue de tourner jusqu'à ce qu'il ait fait 60 boucles ou qu'il soit interrompu, pendant ce temps la, on ne peut pas écrire de commandes sur le terminal.

21) Le processus que l'on vient de lancer a le PID 5760 et il a aussi engendre celui de la commande sleep 5775.

22) La

```
    PID TTY          TIME CMD
   4563 pts/0    00:00:00 bash
   5720 pts/0    00:00:00 ps
22113572t@ST-2019003175:~/Tp2$ ./testboucle.sh&
[1] 5760
22113572t@ST-2019003175:~/Tp2$ Tour de boucle n° 1
Attente d'une seconde…
Tour de boucle n° 2
Attente d'une seconde…
psTour de boucle n° 3
Attente d'une seconde…
Tour de boucle n° 4
Attente d'une seconde…

    PID TTY          TIME CMD
   4563 pts/0    00:00:00 bash
   5760 pts/0    00:00:00 bash
   5775 pts/0    00:00:00 sleep
   5778 pts/0    00:00:00 ps
22113572t@ST-2019003175:~/Tp2$ Tour de boucle n° 5
Attente d'une seconde…
Tour de boucle n° 6
```

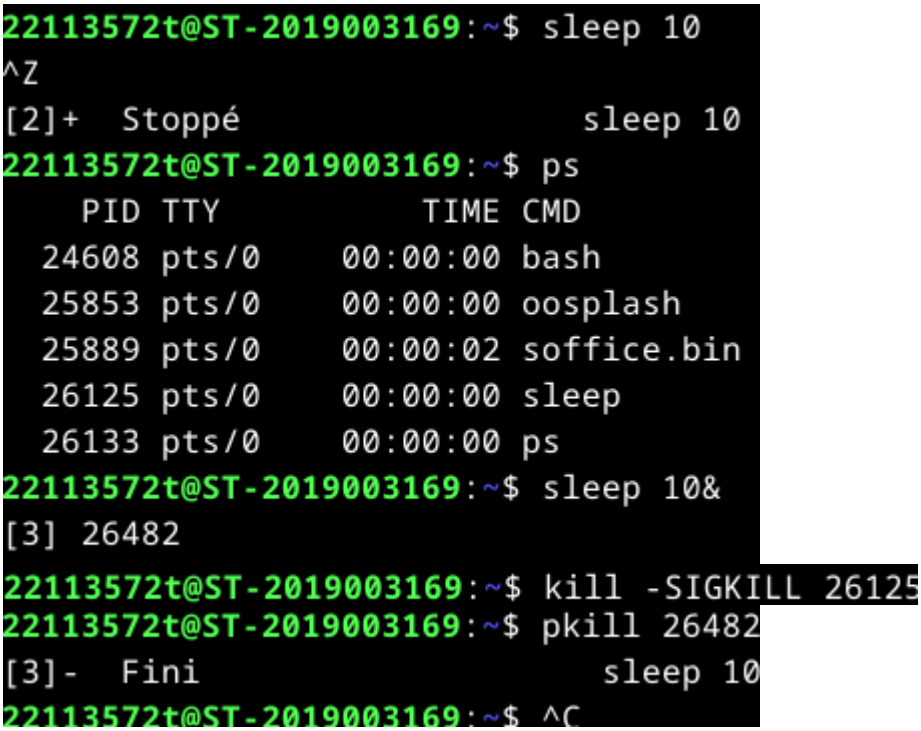
commande « nohup » permet de ne pas afficher la sortie standard d'un programme sur le terminal.

23) En lançant cette commande le terminal exécute le programme qui suit le « nohup » mais aucune commande ne peut être rentrée car le programme n'est pas effectué en arrière plan.

25) Les caractères + et – obtenus sont les processus termines et démarrés.

```
22113572t@ST-2019003175:~/Tp2$ nohup ./testboucle.sh&
[1] 7801
22113572t@ST-2019003175:~/Tp2$ nohup: les entrées son
 ajoutée à 'nohup.out'
sleep 50&
[2] 7822
22113572t@ST-2019003175:~/Tp2$ sleep 30&
[3] 7840
22113572t@ST-2019003175:~/Tp2$ sleep 10&
[4] 7862
22113572t@ST-2019003175:~/Tp2$ ps
    PID TTY          TIME CMD
   4563 pts/0    00:00:00 bash
   7801 pts/0    00:00:00 sh
   7822 pts/0    00:00:00 sleep
   7984 pts/0    00:00:00 sleep
   7987 pts/0    00:00:00 ps
[3]-  Fini                  sleep 30
[4]+  Fini                  sleep 10
```

S




26) Le signal par défaut envoyé par la commande « kil » est un SIGTERM

27) SIGQUIT : fait quitter tous les processus associe au terminal
SIGKILL : termine un processus désigné
SIGCHLD : signal envoyé quand un processus meurt a son parent(systemd est celui auquel l'enfant est rattache automatiquement lorsque son parent meurt)
SIGSTOP : signal envoyé par un processus pour en interrompre un autre

28) On utilise les commandes suivantes et les deux premiers terminaux se ferment, cependant le dernier reste ouvert mais n'est plus utilisable.