Processus : Exercices

Exercice 1 :

Le but de cet exercice est d'observer la mise en concurrence des processus en uilisant le programme python ci-dessous:

```
from os import getpid

p=getpid()

with open("test.txt", "a") as fichier:
    for i in range(1000):
        fichier.write(str(p)+' : '+str(i)+"\n")
        fichier.flush()
```

Quelques explications :

- Ce programme importe la fonction `getpid` du module `os` . Elle renvoie l'identifiant du processus qui sera attribué par le système à l'éxécution de ce programme.
- Le programme utilise ensuite un fichier appelé `test.txt` situé dans le même répertoire que le programme. Ce fichier est créé s'il n'existe pas, et sinon les informations sont ajoutées à la suite dans ce fichier (`a` pour append: ajouter).

1. Recopier le code de ce programme dans un fichier qui sera nommé `pid.py` (par exemple avec IDLE).
2. Exécuter ce programme : Que contient le fichier `test.txt` ?

Reponse:

-
-
-
-

1. Effacer le fichier `test.txt` .
2. Pour observer la mise en concurrence des processus, nous allons éxécuter trois fois ce programme simultanément. Voici la marche à suivre dans un environnement Windows.
   - Créer un nouveau document texte.
   - Y inscrire la suite d'instructions ci-dessous :

     ```
     start /b python pid.py
     start /b python pid.py
     start /b python pid.py
     ```

   - Sauvegarder ce fichier sous le nom `start.bat` (penser à modifier l'extension `.txt` en `.bat`).

1. Le fichier `start.bat` est éxécutable, double cliquer dessus.
2. Décrire ce contient alors le fichier `test.txt` après l'éxécution du fichier.

Réponse:

-
-
-
-
-

1. Renommer le fichier `test.txt` , en `test1.txt` , pour le conserver.
2. Relancer `start.bat` .
3. Le fichier `test.txt` contient-il les mêmes informations que le fichier `test1.txt` ?

Réponse:

-
-
-
-

Remarque :

- Dans un environnement Linux, le comportement de l'ordonnanceur su système est similaire. Pour réalliser ce test, on pourra utiliser la commande suivante dans un terminal pour exécuter trois fois simultanément le même programme:

```
python pid.py & python pid.py & python pid.py
```

Exercice 2 :

Ci-contre, une capture d'écran des principaux processus présents dans un environnement Linux à un instant donné, obtenue àprès l'éxécution de la commande `top` .

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
724	**tlenne**	**20**	**0**	**425604**	**25420**	**16596**	**R**	**28,2**	**2,5**	**0:18.33**	**xfce4-t+**
1018	**tlenne**	**20**	**0**	**19480**	**8572**	**5684**	**R**	**28,2**	**0,8**	**0:17.67**	**python3**
1003	tlenne	20	0	209988	33804	20516	S	8,6	3,3	0:04.07	RDD Pro+
899	tlenne	20	0	2840048	248468	85812	S	7,0	24,6	0:19.65	Web Con+
728	**tlenne**	**20**	**0**	**2920620**	**179780**	**68876**	**R**	**6,3**	**17,8**	**0:13.79**	**firefox+**
692	tlenne	20	0	68772	15408	12700	S	0,7	1,5	0:00.89	xfwm4
809	tlenne	9	-11	649968	15460	10224	S	0,7	1,5	0:00.90	pulseau+
1034	**tlenne**	**20**	**0**	**11140**	**3364**	**2776**	**R**	**0,3**	**0,3**	**0:00.06**	**top**
615	tlenne	20	0	21284	6516	5392	S	0,0	0,6	0:00.04	systemd
616	tlenne	20	0	23248	984	0	S	0,0	0,1	0:00.00	(sd-pam)
631	tlenne	20	0	2388	1244	1244	S	0,0	0,1	0:00.00	sh
639	tlenne	20	0	9320	3868	3028	S	0,0	0,4	0:00.11	dbus-da+
662	tlenne	20	0	5852	0	0	S	0,0	0,0	0:00.00	ssh-age+
672	tlenne	20	0	214512	12456	10324	S	0,0	1,2	0:00.08	xfce4-s+
673	tlenne	20	0	312456	5288	4572	S	0,0	0,5	0:00.00	at-spi-+
678	tlenne	20	0	8968	3444	2996	S	0,0	0,3	0:00.02	dbus-da+
683	tlenne	20	0	174140	5444	4772	S	0,0	0,5	0:00.03	at-spi2+

Pour répondre aux questions, on pourra se documenter sur la commande `kill`

1. Quel est l'identifiant du processus python ?
2. Quel pourcentage de la puissance du processeur occupe-t-il ?
3. Quel est le PID du processus occupe le plus la mémoire ?
4. A quel processus correspond le PID 631 ?
5. Combien de processus sont en cours d'éxécution ?
6. Le processus `web content` est planté, mais l'utilisateur veut essayer de l'arrêter proprement pour ne pas perdre son état. Quelle commande peut-il saisir ?
7. Que se passe-t-il si l'on exécute `kill -9 728` ?
8. Que se passe-t-il si l'on exéctue `kill 1034` ?

Réponses:

1.
2.
3.
4.
5.
6.
7.
8.

Remaque :

Il existe des émulateurs de terminaux Linux en ligne qui permettent de tester les commandes :

- https://bellard.org/jslinux/vm.html?cpu=riscv64&url=fedora29-riscv-2.cfg&mem=256 (https://bellard.org/jslinux/vm.html?cpu=riscv64&url=fedora29-riscv-2.cfg&mem=256)
- https://www.offidocs.com/jor1k/demos/main.html?user=thomas&cpu=asm&n=1&relayURL=wss%3A%2F%2Frelay.widgetry.org%2F (https://www.offidocs.com/jor1k/demos/main.html?user=thomas&cpu=asm&n=1&relayURL=wss%3A%2F%2Frelay.widgetry.org%2F)

Exercice 3:

La circulation est impossible pour les quatre véhicules ci-contre car chacun veut aller tout droit et doit laisser la priorité à droite.

En identifiant les voitures à des processus et les portions de route à des ressources, montrer que les quatre conditions de Coffman sont réunies et qu'il s'agit donc d'une situation d'interblocage.

On notera V1, V2 V3 et V4 les processus(les voitures) et R1, R2, R3, R4 les ressources(les portions de route devant les voitures):

- V1 est la voiture rouge et veut passer sur la route R1.
- V2 est la voiture jaune et veut passer sur la route R2.
- V3 est la voiture verte et veut passer sur la route R3.
- V4 est la voiture bleue et veut passer sur la route R4.

Réponse :

1. Les ressources sont en accès exclusifs :
2. Rétention et attente :
3. Non préemption :
4. Attente cyclique :