

EXERCICE 4 (4 points)

Cet exercice porte sur les systèmes d'exploitation : gestion des processus et des ressources.

Les parties A et B peuvent être traitées indépendamment.

Partie A :

Dans un bureau d'architectes, on dispose de certaines ressources qui ne peuvent être utilisées simultanément par plus d'un processus, comme l'imprimante, la table traçante, le modem. Chaque programme, lorsqu'il s'exécute, demande l'allocation des ressources qui lui sont nécessaires. Lorsqu'il a fini de s'exécuter, il libère ses ressources.

<u>Programme 1</u>	<u>Programme 2</u>	<u>Programme 3</u>
demander (table traçante)	demander (modem)	demander (imprimante)
demander (modem)	demander (imprimante)	demander (table traçante)
exécution	exécution	exécution
libérer (modem)	libérer (imprimante)	libérer (table traçante)
libérer (table traçante)	libérer (modem)	libérer (imprimante)

On appelle p1, p2 et p3 les processus associés respectivement aux programmes 1, 2 et 3.

1. Les processus s'exécutent de manière concurrente.
Justifier qu'une situation d'interblocage peut se produire.
2. Modifier l'ordre des instructions du programme 3 pour qu'une telle situation ne puisse pas se produire. Aucune justification n'est attendue.
3. Supposons que le processus p1 demande la table traçante alors qu'elle est en cours d'utilisation par le processus p3. Parmi les états suivants, quel sera l'état du processus p1 tant que la table traçante n'est pas disponible :
a) élu b) bloqué c) prêt d) terminé

Partie B :

Avec une ligne de commande dans un terminal sous Linux, on obtient l'affichage suivant :

UID	PID	PPID	C	STIME	TTY	TIME	CMD
...							
pi	6211	831	8	09:07	?	00:01:16	/usr/lib/chromium-browser/chromium-browser-v7 --disable-quic --enable-tcp-fast-open --p
pi	6252	6211	0	09:07	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/lib
pi	6254	6252	0	09:07	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/lib
pi	6294	6211	4	09:07	?	00:00:40	/usr/lib/chromium-browser/chromium-browser-v7 --type=gpu-process --field-trial-handle=1
pi	6300	6211	1	09:07	?	00:00:16	/usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758
pi	6467	6254	1	09:07	?	00:00:11	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	11267	6254	2	09:12	?	00:00:15	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	12035	836	0	09:13	?	00:00:00	/usr/lib/libreoffice/program/oosplash --writer file:///home/pi/Desktop/mon_fichier.odt
pi	12073	12035	2	09:13	?	00:00:15	/usr/lib/libreoffice/program/soffice.bin --writer file:///home/pi/Desktop/mon_fichier.c
pi	12253	831	1	09:13	?	00:00:07	/usr/bin/python3 /usr/bin/sense_emu_gui
pi	20010	6211	1	09:21	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=10758
pi	20029	6254	56	09:21	?	00:00:28	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20339	6254	4	09:21	?	00:00:01	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20343	6254	2	09:21	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20464	6211	17	09:22	?	00:00:00	/proc/self/exe --type=utility --field-trial-handle=1075863133478894917,6306120996223181
pi	20488	6254	14	09:22	?	00:00:00	/usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi	20519	676	0	09:22	pts/0	00:00:00	ps -ef

La documentation Linux donne la signification des différents champs :

- UID : identifiant utilisateur effectif ;
- PID : identifiant de processus ;
- PPID : PID du processus parent ;
- C : partie entière du pourcentage d'utilisation du processeur par rapport au temps de vie des processus ;
- STIME : l'heure de lancement du processus ;
- TTY : terminal de contrôle
- TIME : temps d'exécution
- CMD : nom de la commande du processus

1. Parmi les quatre commandes suivantes, laquelle a permis cet affichage ?

- a) `ls -l`
- b) `ps -ef`
- c) `cd ..`
- d) `chmod 741 processus.txt`

2. Quel est l'identifiant du processus parent à l'origine de tous les processus concernant le navigateur Web (chromium-browser) ?

3. Quel est l'identifiant du processus dont le temps d'exécution est le plus long ?

EXERCICE 5 (4 points)

Cet exercice porte sur les structures de données linéaires

Une méthode simple pour gérer l'ordonnancement des processus est d'exécuter les processus en une seule fois et dans leur ordre d'arrivée.

- Parmi les propositions suivantes, quelle est la structure de données la plus appropriée pour mettre en œuvre le mode FIFO (First In First Out) ?
 - liste
 - dictionnaire
 - pile
 - file
- On choisit de stocker les données des processus en attente à l'aide d'une liste Python `lst`. On dispose déjà d'une fonction `retirer(lst)` qui renvoie l'élément `lst[0]` puis le supprime de la liste `lst`. Écrire en Python le code d'une fonction `ajouter(lst, proc)` qui ajoute à la fin de la liste `lst` le nouveau processus en attente `proc`.

On choisit maintenant d'implémenter une file `file` à l'aide d'un couple $(p1, p2)$ où `p1` et `p2` sont des piles. Ainsi `file[0]` et `file[1]` sont respectivement les piles `p1` et `p2`.

Pour enfiler un nouvel élément `elt` dans `file`, on l'empile dans `p1`.

Pour défiler `file`, deux cas se présentent.

- La pile `p2` n'est pas vide : on dépile `p2`.
- La pile `p2` est vide : on dépile les éléments de `p1` en les empilant dans `p2` jusqu'à ce que `p1` soit vide, puis on dépile `p2`.

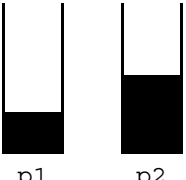
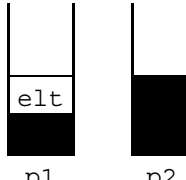
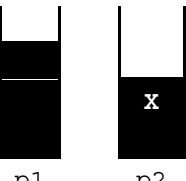
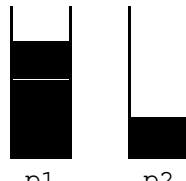
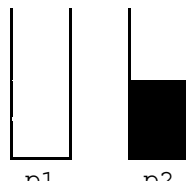
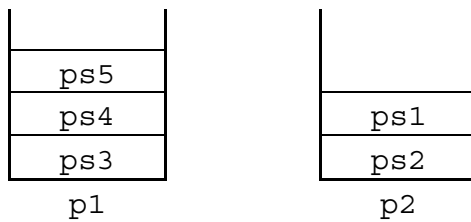
	État de la file avant	État de la file après
<code>enfiler(file, elt)</code>	 p1 p2	 p1 p2
<code>defiler(file)</code> cas où <code>p2</code> n'est pas vide	 p1 p2	 p1 p2
<code>defiler(file)</code> cas où <code>p2</code> est vide	 p1 p2	 p1 p2

Illustration du fonctionnement des fonctions `enfiler` et `défiler`.

3. On considère la situation représentée ci-dessous.



On exécute la séquence d'instructions suivante :

```
enfiler(file, ps6)
defiler(file)
defiler(file)
defiler(file)
enfiler(file, ps7)
```

Représenter le contenu final des deux piles à la suite de ces instructions.

4. On dispose des fonctions :

- `empiler(p, elt)` qui empile l'élément `elt` dans la pile `p`,
- `depiler(p)` qui renvoie le sommet de la pile `p` si `p` n'est pas vide et le supprime,
- `pile_vide(p)` qui renvoie `True` si la pile `p` est vide, `False` si la pile `p` n'est pas vide.

a. Écrire en Python une fonction `est_vide(f)` qui prend en argument un couple de piles `f` et qui renvoie `True` si la file représentée par `f` est vide, `False` sinon.

b. Écrire en Python une fonction `enfiler(f, elt)` qui prend en arguments un couple de piles `f` et un élément `elt` et qui ajoute `elt` en queue de la file représentée par `f`.

c. Écrire en Python une fonction `defiler(f)` qui prend en argument un couple de piles `f` et qui renvoie l'élément en tête de la file représentée par `f` en le retirant.