

Compte rendu

TP7

Accès au fichiers avec mmap ;

Partage de mémoire entre processus ;

Synchronisation entre processus via sémaphores

Ex. 1. Accès en lecture/écriture avec mmap

Principe : On souhaite faire un programme capable de prendre le nom d'un fichier binaire en argument comportant une série d'entiers sur 16 bits représentés en little-endian et le convertit en big endian. Pour ce faire on va mapper le fichier en mémoire, la série d'entiers sera donc manipulable comme un tableau. Pour la conversion, il faudra échanger les entiers deux par deux (1^{er} devient seconde et second devient 1^{er} même chose pour le troisième et le quatrième).

Vérification :

```
[0]utilisateur@Ubuntu:~$ od --format=x1 test.bin
00000000 12 34 56 78
00000004
[0]utilisateur@Ubuntu:~$ python3 TP7exo1.py test.bin
0x34
0x12
0x78
0x56
[0]utilisateur@Ubuntu:~$ od --format=x1 test.bin
00000000 34 12 78 56
00000004
```

On obtient bien ce que l'on souhaite cad une série d'entier en big-endian .

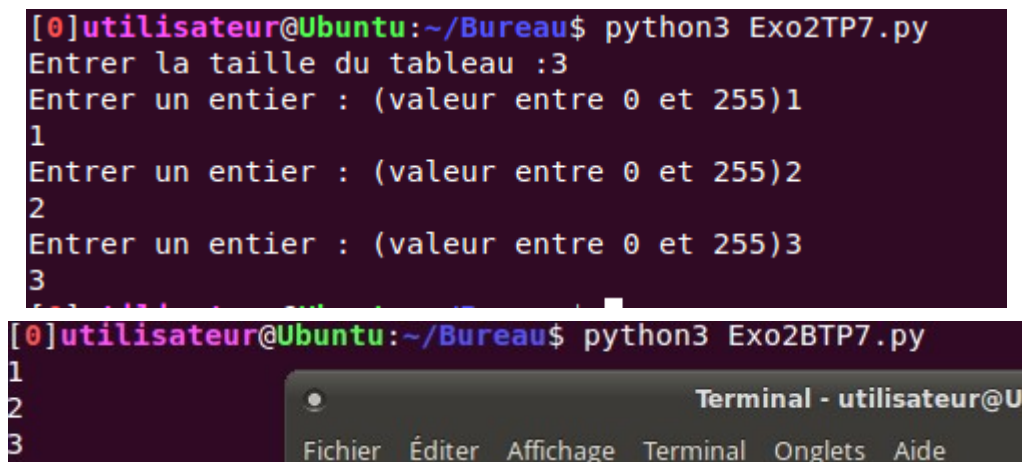
Ex. 2. Communication par tube nommé

Principe : On souhaite faire communiquer deux programmes par mémoire partagée. On utilisera **SharedMemory** pour créer et accéder à la mémoire partagée. Le premier va demander à l'utilisateur d'entrer un nombre d'entier voulu et de les saisir un par un. Ces valeurs seront donc stockées dans la mémoire partagée grâce avec **mmap** et on y mettra d'abord la nombre d'entiers entrés par l'utilisateur et ensuite les entiers saisis. L'autre programme lui accèdera à la mémoire partagée et l'affichera, grâce au **mmap** on pourra utiliser cette mémoire comme un tableau.

Vérification :

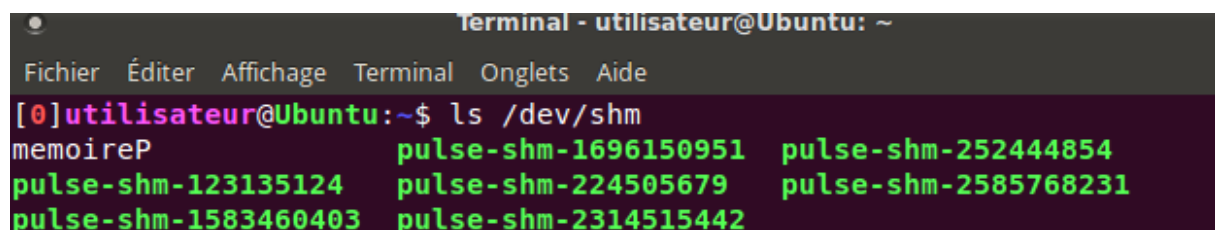
```
[0]utilisateur@Ubuntu:~/Bureau$ python3 Exo2TP7.py
Entrer la taille du tableau :3
Entrer un entier : (valeur entre 0 et 255)1
1
Entrer un entier : (valeur entre 0 et 255)2
2
Entrer un entier : (valeur entre 0 et 255)3
3
```

```
[0]utilisateur@Ubuntu:~/Bureau$ python3 Exo2BTP7.py
1
2
3
```



On arrive donc à faire communiquer les deux programmes par mémoire partagée.

Question : lister le contenu du répertoire `/dev/shm` pendant l'exécution de P1. Que concluez-vous ?



```
terminal - utilisateur@Ubuntu: ~
Fichier Éditer Affichage Terminal Onglets Aide
[0]utilisateur@Ubuntu:~$ ls /dev/shm
memoireP          pulse-shm-1696150951  pulse-shm-252444854
pulse-shm-123135124 pulse-shm-224505679  pulse-shm-2585768231
pulse-shm-1583460403 pulse-shm-2314515442
```

La mémoire partagée s'y trouve.

Ex. 3. Synchronisation de processus de par sémaphore

Principe : On souhaite améliorer l'exercice précédent. On veut maintenant que le programme qui affiche la série d'entiers soit lancé en premier et qu'il attende que l'utilisateur ait fini d'entrer les entiers dans le premier programme. Pour ce faire on utilisera un sémaphore et nos souvenirs du cours magistral.

Vérification :

```
[0]utilisateur@Ubuntu:~/Bureau$ python3 Exo3TP7.py
Entrer la taille du tableau :3
Entrer un entier : (valeur entre 0 et 255)1
1
Entrer un entier : (valeur entre 0 et 255)2
2
Entrer un entier : (valeur entre 0 et 255)3
3

[0]utilisateur@Ubuntu:~/Bureau$ python3 Exo3BTP7.py
1
2
3
```

On a donc bien ce que l'on souhaite cad que le second programme attend que l'utilisateur ai saisie tout les entiers.