Nom : Laura BINACCHI

29 avril 2021 : labo A

Modifications du code reader-writer :

- suppression de la priorité par programmation (listings 4.19-21, pp. 79-81)
- ajout de paramètres au lancement du programme (readers and writers niceness)
- ajout des fichiers statistics.h et statistics.c pour logger des informations dans des fichiers csv
- déplacement du sleep dans les fonctions de lecture et d'écriture plutôt que dans les fonctions des threads

Structure:

```
- exa-readers-writers-nice
      + build
      + include
      + src
      CMakeLists.txt
```

Compilation:

```
~/2021_TR/exa-readers-writers-nice/build
      cmake -DCMAKE_C_FLAGS=-g ..
~/2021_TR/exa-readers-writers-nice/build
      make
```

-> création de l'exécutable **`reader-writer`** dans le répertoire **`build`**

Lancement du programme :

```
./reader_writer readers readers_niceness writers
      writers_niceness max_value
```

- **`readers`** : nombre de threads lecteurs
- **`readers_niceness`** : valeur de courtoisie des threads lecteurs
- **`writers`** : nombre de threads écrivains
- **`writers_niceness`** : valeur de courtoisie des threads écrivains
- **`max_value`** : valeur maximale de la donnée

Les fichiers csv de statistiques sont créés dans le sous-répertoire **`stats`** relatif au répertoire depuis lequel est lancé le programme.

Le programme produit les mêmes affichages que l'exercice 3.

Comparaison avec des valeurs de nice différentes : on voit bien les writers rassemblés au début quand les readers sont plus courtois (c'est toujours plus ou moins le même type de répartition -> à vérifier en faisant des stats).

```
~/2021_TR/exa-readers-writers-nice/build$
./reader_writer 20 -20 5 19 8
reader   1 reads        0
writer   1 writes   1 TOTAL>   1
writer   1 writes   1 TOTAL>   2
reader   2 reads        2
writer   3 writes   3 TOTAL>   5
reader   3 reads        5
writer   4 writes   4 TOTAL>   8
reader   4 reads        8
writer   5 writes   5 TOTAL>   8
reader  20 reads        8
reader  12 reads        8
reader  17 reads        8
reader  18 reads        8
reader   9 reads        8
reader  13 reads        8
reader   5 reads        8
reader   6 reads        8
reader   1 reads        8
reader  16 reads        8
reader   8 reads        8
reader  10 reads        8
reader   7 reads        8
reader  15 reads        8
reader  11 reads        8
reader  19 reads        8
reader  14 reads        8
writer   2 writes   2 TOTAL>   8
reader   2 reads        8
writer   1 writes   1 TOTAL>   8
writer   3 writes   3 TOTAL>   8
reader   3 reads        8
```

```
~/2021_TR/exa-readers-writers-nice/build$
./reader_writer 20 19 5 -20 8
writer   1 writes   1 TOTAL>   1
writer   1 writes   1 TOTAL>   2
writer   1 writes   1 TOTAL>   3
writer   1 writes   1 TOTAL>   4
writer   1 writes   1 TOTAL>   5
writer   1 writes   1 TOTAL>   6
writer   1 writes   1 TOTAL>   7
writer   1 writes   1 TOTAL>   8
writer   3 writes   3 TOTAL>   8
reader   1 reads        8
reader   3 reads        8
writer   2 writes   2 TOTAL>   8
writer   4 writes   4 TOTAL>   8
writer   5 writes   5 TOTAL>   8
reader   5 reads        8
reader   4 reads        8
reader   6 reads        8
reader   2 reads        8
reader   9 reads        8
reader   8 reads        8
reader  10 reads        8
reader  11 reads        8
reader   7 reads        8
reader  14 reads        8
reader  16 reads        8
reader  17 reads        8
reader  15 reads        8
reader  12 reads        8
reader  13 reads        8
reader  18 reads        8
reader  19 reads        8
reader  20 reads        8
```

1er mai 2021 : labo A

Modification des printf pour plus de lisibilité :

```
r   1 <   0
w   3 >   3 (+3)
r   7 <   3
r  10 <   3
r   3 <   3
w   5 >   8 (+5)
```

Vérification des courtoisies des threads avec les printf

```
printf("w%d        nice        %d\n",        params.reader_id,
getpriority(PRIO_PROCESS, 0));
```

et

```
printf("w%d        nice        %d\n",        params.writer_id,
getpriority(PRIO_PROCESS, 0));
```

ajoutés dans les threads lecteurs et écrivains (read_thread.c et write_thread.c)

Lancement du programme avec une courtoisie minimale pour les écrivains et maximale pour les lecteurs : **`./reader_writer 10 19 5 -20 10`**
résultat :

```
w1 nice 0
r1 nice 19
```

la même commande avec sudo:

```
w1 nice -20
```

-> attention en le lançant car le fichier de stats prend en compte le paramètre du programme et pas la valeur réelle du nice.

Ajout d'une barrière pour pouvoir supprimer la temporisation dans les codes des threads sans que le premier thread lecteur ou écrivain fasse (presque) tout le travail.

- ajout d'une barrière (pthread_barrier_t *barrier) dans la paramètres des threads lecteurs et écrivains (read_thead.h et write_thead.h)
- initialisation de la barrière dans le main
- wait sur la barrière dans les codes des lecteurs et écrivains avant de lancer la boucle de lecture ou d'écriture

Résultat : le premier thread passe quand même plusieurs fois en premier en supprimant le sleep. Cela peut s'expliquer par le fait qu'il soit le premier thread réveillé par le kernel puisqu'il est le premier thread à avoir été bloqué et qu'il a la même priorité que les autres écrivains et que la VM n'a qu'un CPU. En mettant des valeurs de courtoisies différentes aux écrivains, ce sont bien les écrivains avec une valeur de nice négative qui passent avant ceux avec une valeur positive, indépendamment de leur id (i.e. de leur ordre de création).

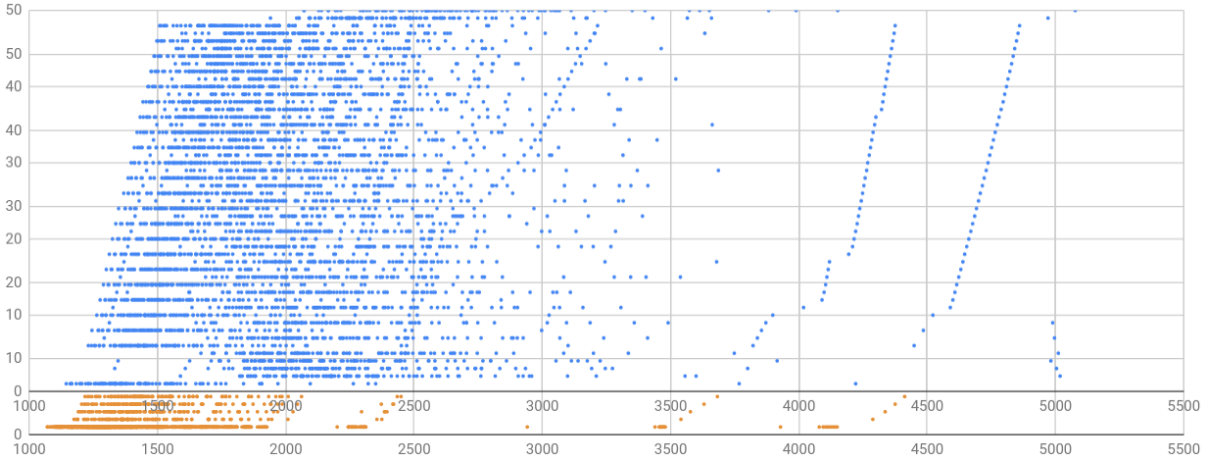
Statistiques :

- création d'un bash pour lancer le programme en boucle (sans oublier le sudo) : tests.sh
- création par le programme de fichiers csv dans le dossier stats : un fichier par type de thread (lecteur ou écrivain), nombre de lecteurs, courtoisie des lecteurs, nombre d'écrivains, courtoisie des écrivains et et valeur maximale -> avec les mêmes paramètres, les données sont ajoutées au fichier csv plutôt que de créer un nouveau fichier à chaque lancement du programme

Après plusieurs essais il est apparu que l'efficacité du nice est plus grande avec un petit nombre de threads et une petite valeur maximale. J'ai également enlevé tous les sleep pour avoir le moins de perturbations possibles dans les statistiques (les sleep masquent par exemple les priorités données au premier lecteur sur les autres dans le cas d'un seul CPU).

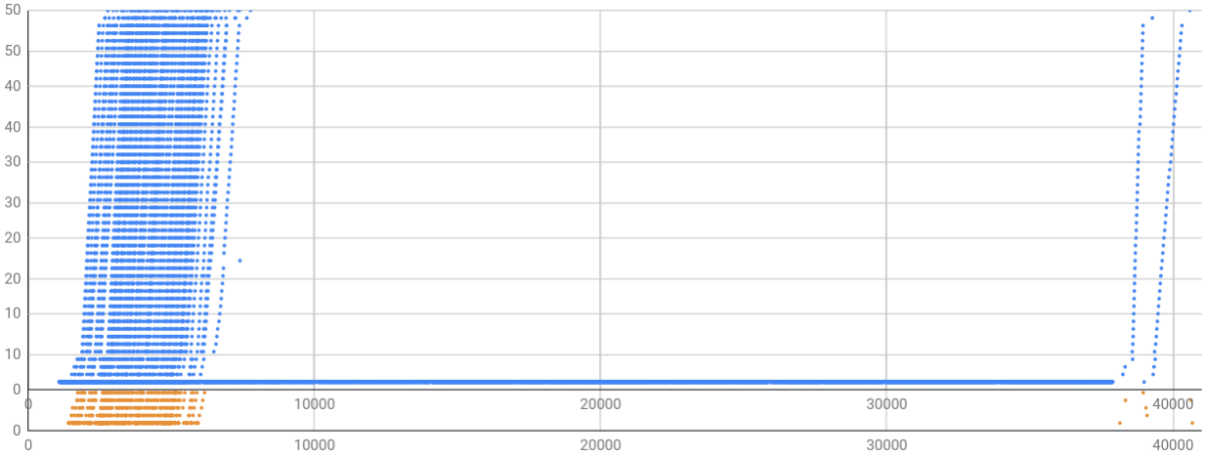
Résultats sur 100 exécutions pour :

- 50 lecteurs avec courtoisie de 19
- 5 écrivains avec courtoisie de -20
- valeur maximale de 10



Résultats sur 100 exécutions pour :

- 50 lecteurs avec courtoisie de -20
- 5 écrivains avec courtoisie de 19
- valeur maximale de 10



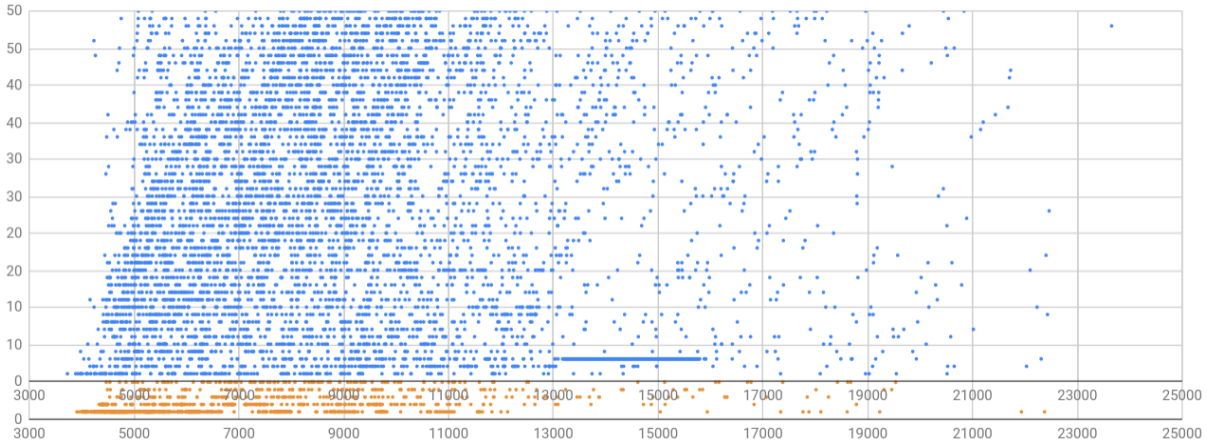
Sur les graphiques, chaque lecture est représentée par un point bleu et chaque écriture par un point orange. L'axe des x représente le temps en cycles CPU écoulé depuis l'ouverture du fichier csv par le programme. L'axe des y représente les ID des threads. Les graphiques des lectures et des écritures ont été superposés pour faciliter la comparaison des résultats.

Sur le premier graphique, les écritures commencent avant les lectures et se terminent plus tôt que sur le second graphique. Sur le premier graphique, c'est surtout le premier écrivain qui travaille et sur le second c'est le premier lecteur. Cet effet peut être atténué par les temporisations ou en ajoutant plus de CPU à la VM mais l'effet du nice n'est alors plus visible.

Mêmes tests avec 4 CPU

- 50 lecteurs de courtoisie 19
- 5 écrivains de courtoisie -20
- valeur maximale de 10

- 50 lecteurs de courtoisie -20
- 5 écrivains de courtoisie -19
- valeur maximale de 10



-> résultats de ces tests dans le dossier tests1

2 mai 2021 : labo A

Les graphiques en box plot permettent mieux de voir la répartition des populations de lecteurs et d'écrivains :
https://en.wikipedia.org/wiki/Box_plot
https://www.statology.org/box-plot-google-sheets/

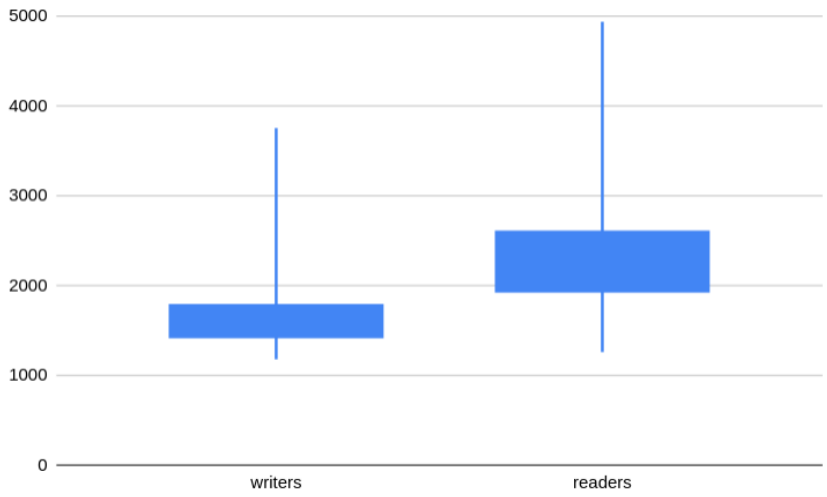
J'ai ressorti des données en faisant des tests avec les sleep mais les résultats ne sont vraiment pas concluants. Je les laisse quand même dans le dossier stats_w_sleep_1CPU où il y a les csv sortis par le programme et la spreadsheet dans laquelle les données ont été importées avec les graphiques générées et calcul des moyennes de lecteurs et d'écrivains : les moyennes et les graphiques sont proches.

J'ai aussi refait une série de tests avec un CPU et sans sleep et en mettant la valeur d'incrément à 1 pour limiter encore la part d'aléatoire (dossier stats_wo_sleep_1CPU).

Paramètres communs :
- 1 CPU
- pas de sleep
- valeur maximale de 10
- valeur d'incrément de 1

Résultats sur 100 exécutions pour :
- 50 lecteurs avec courtoisie de 19
- 10 écrivains avec courtoisie de -20

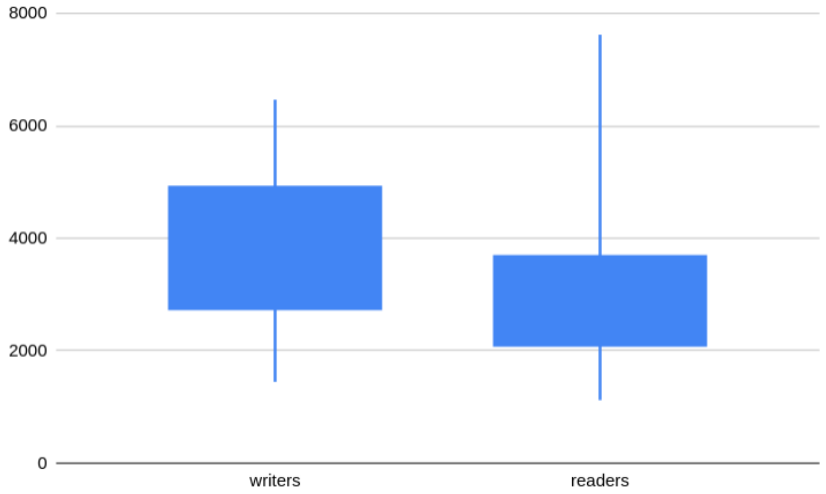
moyenne d'écrivains par exécution : 19
moyenne de lecteurs par exécution : 50



Résultats sur 100 exécutions pour :

- 50 lecteurs avec courtoisie de 0
- 10 écrivains avec courtoisie de 19

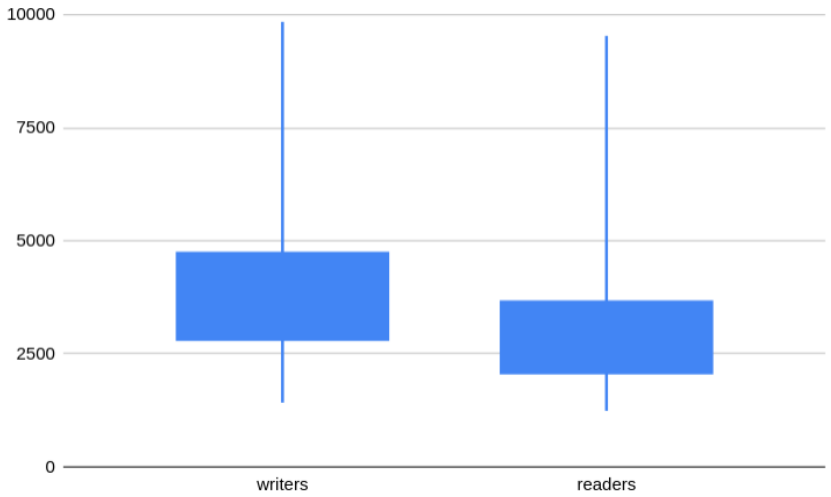
moyenne d'écrivains par exécution : 19
moyenne de lecteurs par exécution : 623.83



Résultats sur 100 exécutions pour :

- 50 lecteurs avec courtoisie de -20
- 10 écrivains avec courtoisie de 19

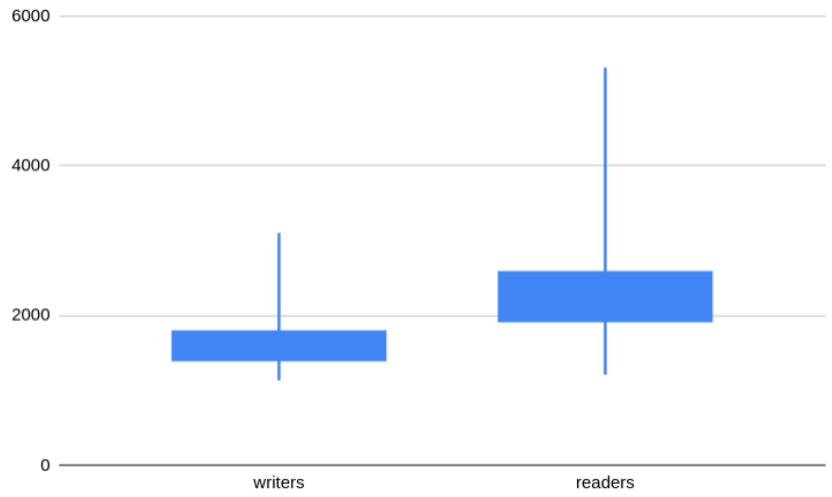
moyenne d'écrivains par exécution : 19
moyenne de lecteurs par exécution : 630.85



Résultats sur 100 exécutions pour :

- 50 lecteurs avec courtoisie de 0
- 10 écrivains avec courtoisie de 0

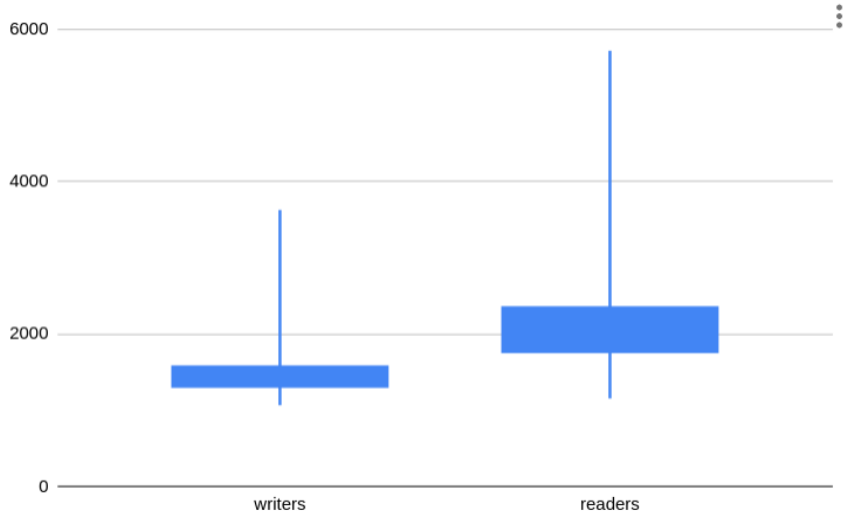
moyenne d'écrivains par exécution : 19
moyenne de lecteurs par exécution : 50



Pour vérification, j'ai lancé le programme sans la commande nice
Résultats sur 100 exécutions pour :

- 50 lecteurs sans courtoisie
- 10 écrivains san courtoisie

moyenne d'écrivains par exécution : 19
moyenne de lecteurs par exécution : 50



Observations :

- l'exécution sans nice et avec nice de 0 donne bien des résultats similaires
- le nombre moyen d'écrivains toujours à 19 : quand le 10e passe pour incrémenter, il s'arrête puis les 9 autres qui sont dans la file refont un passage avant de s'arrêter
- on constate une différence entre les exécutions avec courtoisie supérieure pour les écrivain (courtoisie à 19 contre 0 ou -20 pour les lecteurs)
- par contre il n'y a pas de différence notable entre l'exécution sans courtoisie ou avec courtoisie nulle et celle ou une courtoisie supérieure est donnée au lecteurs : il semblerait que le programme soit déjà favorable aux écrivains (du moins avec les paramètres testés, i.e. peu d'écrivains et beaucoup de lecteurs)

Conclusion : augmenter la priorité des threads écrivains avec les courtoisies fonctionne en augmentant la courtoisie des lecteurs mais pas en diminuant celle des écrivains.

Projet lié : laba_readers_writers.zip

Remarque : pour éviter de devoir lancer le programme avec sudo en autorisant les nice négatifs, il est possible d'autoriser l'exécution des nice pour l'exécutable avec la commande **`sudo setcap cap_sys_nice=eip ./executable`** où executable est le nom du programme. Les sudo peuvent alors être supprimés du bash de test et les fichiers csv et le dossier qui les contient n'appartiennent plus à l'utilisateur root.

8 mai 2021 : labo B

Modification du code reader-writer du labo A (sans priorité par programmation et avec barrière avant le démarrage des threads) :

- scheduler en round robin
- priorité paramétrable au lancement du programme
- sleeps non aléatoires
- valeur d'incrément à 1

Ici l'implémentation de la barrière est vraiment nécessaire puisque les threads lecteurs et écrivains vont avoir une priorité supérieur aux threads de création.

Résultats :

```
sudo ./reader_writer 15 1 5 2 10
```

```
w   1 >   1 (+1)       w   1 >  10 (+1)       r   6 <  10
w   2 >   2 (+1)       w   2 >  10 (+1)       r   7 <  10
w   3 >   3 (+1)       w   3 >  10 (+1)       r   8 <  10
w   4 >   4 (+1)       w   4 >  10 (+1)       r   9 <  10
w   5 >   5 (+1)       r  15 <  10            r  10 <  10
w   1 >   6 (+1)       r   1 <  10            r  11 <  10
w   2 >   7 (+1)       r   2 <  10            r  12 <  10
w   3 >   8 (+1)       r   3 <  10            r  13 <  10
w   4 >   9 (+1)       r   4 <  10            r  14 <  10
w   5 >  10 (+1)       r   5 <  10
```

```
sudo ./reader_writer 15 1 5 1 10
```

```
r  15 <   0            r  10 <   5            r   5 <  10
w   1 >   1 (+1)       r  11 <   5            r   6 <  10
w   2 >   2 (+1)       r  12 <   5            r   7 <  10
w   3 >   3 (+1)       r  13 <   5            r   8 <  10
w   4 >   4 (+1)       r  14 <   5            r   9 <  10
w   5 >   5 (+1)       r  15 <   5            r  10 <  10
r   1 <   5            w   1 >   6 (+1)       r  11 <  10
r   2 <   5            w   2 >   7 (+1)       r  12 <  10
r   3 <   5            w   3 >   8 (+1)       r  13 <  10
r   4 <   5            w   4 >   9 (+1)       r  14 <  10
r   5 <   5            w   5 >  10 (+1)       r  15 <  10
r   6 <   5            r   1 <  10            w   1 >  10 (+1)
r   7 <   5            r   2 <  10            w   2 >  10 (+1)
r   8 <   5            r   3 <  10            w   3 >  10 (+1)
r   9 <   5            r   4 <  10            w   4 >  10 (+1)
```

```
sudo ./reader_writer 15 2 5 1 10
```

```
r  15 <   0          r   9 <   3          r   8 <   7
r   1 <   0          r  10 <   3          r   9 <   7
r   2 <   0          r  14 <   3          r  10 <   7
r   3 <   0          w   4 >   4 (+1)     r  11 <   7
r   4 <   0          r  15 <   4          r  12 <   7
r   5 <   0          r   1 <   4          r  13 <   7
r   6 <   0          r   2 <   4          r  14 <   7
r   7 <   0          r   3 <   4          w   3 >   8 (+1)
r   8 <   0          r   4 <   4          r  15 <   8
r   9 <   0          r   5 <   4          r   1 <   8
r  10 <   0          r   6 <   4          r   2 <   8
r  11 <   0          r   7 <   4          r   3 <   8
r  12 <   0          r   8 <   4          r   4 <   8
r  13 <   0          r   9 <   4          r   5 <   8
r  14 <   0          r  10 <   4          r   6 <   8
w   1 >   1 (+1)     r  11 <   4          r   7 <   8
r  15 <   1          r  12 <   4          r   8 <   8
r   1 <   1          r  13 <   4          r   9 <   8
r   2 <   1          r  14 <   4          r  10 <   8
r   3 <   1          w   5 >   5 (+1)     r  11 <   8
r   4 <   1          r  15 <   5          r  12 <   8
r   5 <   1          r   1 <   5          r  13 <   8
r   6 <   1          r   2 <   5          r  14 <   8
r   7 <   1          r   3 <   5          w   4 >   9 (+1)
r   8 <   1          r   4 <   5          r  15 <   9
r   9 <   1          r   5 <   5          r   1 <   9
r  10 <   1          r   6 <   5          r   2 <   9
r  11 <   1          r   7 <   5          r   3 <   9
r  12 <   1          r   8 <   5          r   4 <   9
r  13 <   1          r   9 <   5          r   5 <   9
r  14 <   1          r  10 <   5          r   6 <   9
w   2 >   2 (+1)     r  11 <   5          r   7 <   9
r  15 <   2          r  12 <   5          r   8 <   9
r   1 <   2          r  13 <   5          r   9 <   9
r   2 <   2          r  14 <   5          r  10 <   9
r   3 <   2          w   1 >   6 (+1)     r  11 <   9
r   4 <   2          r  15 <   6          r  12 <   9
r   5 <   2          r   1 <   6          r  13 <   9
r   6 <   2          r   2 <   6          r  14 <   9
r   7 <   2          r   3 <   6          w   5 >  10 (+1)
r   8 <   2          r   4 <   6          r  15 <  10
r   9 <   2          r   5 <   6          r   1 <  10
r  10 <   2          r   6 <   6          r   2 <  10
r  11 <   2          r   7 <   6          r   3 <  10
r  12 <   2          r   8 <   6          r   4 <  10
r  13 <   2          r   9 <   6          r   5 <  10
r  14 <   2          r  10 <   6          r   6 <  10
w   3 >   3 (+1)     r  11 <   6          r   7 <  10
r  15 <   3          r  12 <   6          r   8 <  10
r   1 <   3          r  13 <   6          r   9 <  10
r   2 <   3          r  14 <   6          r  10 <  10
r   3 <   3          w   2 >   7 (+1)     r  11 <  10
r   4 <   3          r  15 <   7          r  12 <  10
r   5 <   3          r   1 <   7          r  13 <  10
```

```
r   6 <   3          r   2 <   7          r  14 <  10
r   7 <   3          r   3 <   7          w   1 >  10 (+1)
r   8 <   3          r   4 <   7          w   2 >  10 (+1)
r  11 <   3          r   5 <   7          w   3 >  10 (+1)
r  12 <   3          r   6 <   7          w   4 >  10 (+1)
r  13 <   3          r   7 <   7
```

Sans priorité (lancement des threads avec NULL au lieu des attributs définis) :
`./reader_writer 15 - 5 - 10`

```
r   1 <   0          r   9 <   5          r   6 <  10
w   2 >   1 (+1)     r  11 <   5          r  10 <  10
r   2 <   1          r   5 <   5          r   7 <  10
w   3 >   2 (+1)     r  14 <   5          r  13 <  10
r   3 <   2          w   1 >   6 (+1)     r  11 <  10
w   4 >   3 (+1)     r   1 <   6          r  14 <  10
r   4 <   3          w   2 >   7 (+1)     r  12 <  10
w   5 >   4 (+1)     r   2 <   7          w   1 >  10 (+1)
r   8 <   4          w   3 >   8 (+1)     r   1 <  10
r   6 <   4          r   3 <   8          r  15 <  10
r   7 <   4          w   4 >   9 (+1)     w   2 >  10 (+1)
r   5 <   4          r   4 <   9          r   2 <  10
w   1 >   5 (+1)     w   5 >  10 (+1)     w   3 >  10 (+1)
r  12 <   5          r   8 <  10          r   3 <  10
r  10 <   5          r   9 <  10          w   4 >  10 (+1)
r  13 <   5          r   5 <  10          r   4 <  10
```

Les résultats semblent assez proche du run avec la même priorité pour les lecteurs et les écrivains -> à vérifier avec les statistiques

9 mai 2021 : labo B

Sortie des statistiques en lançant le programme plusieurs fois avec les mêmes paramètres (fichier tests.sh, résultats dans le répertoire stats).

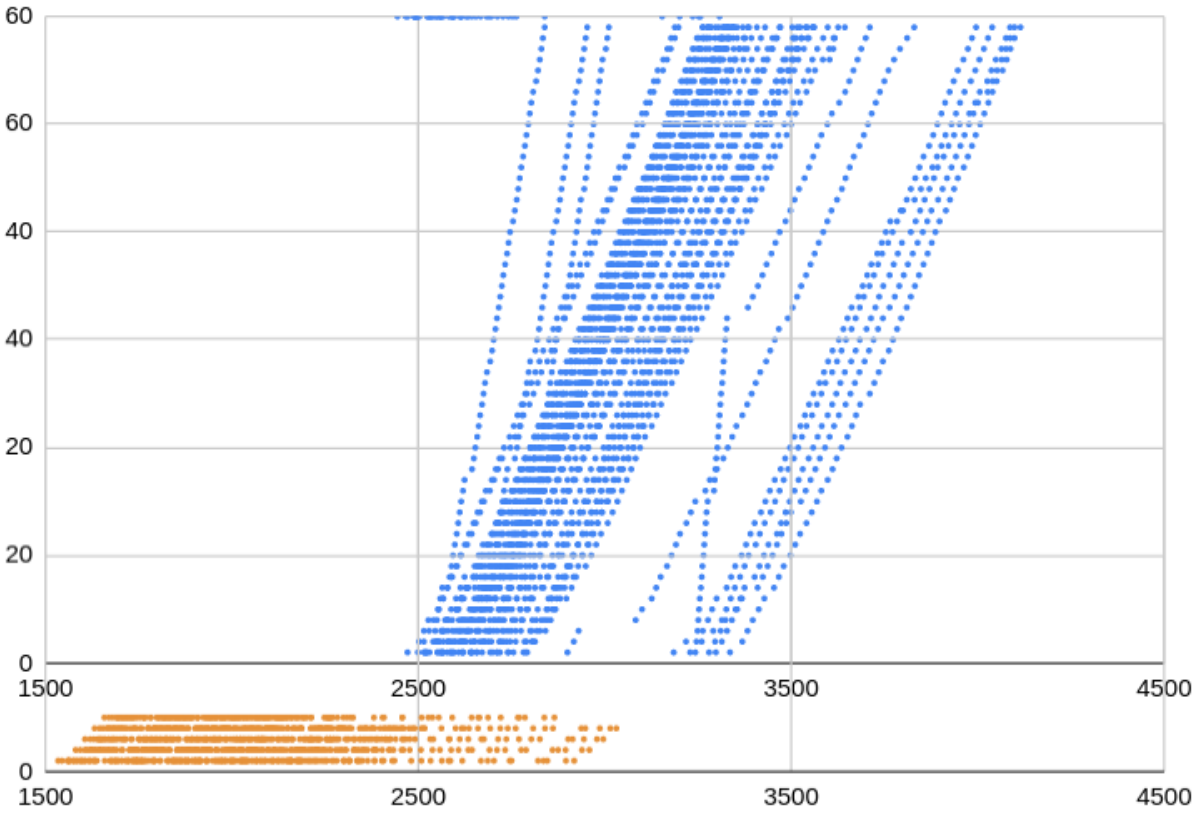
Paramètres communs des tests :

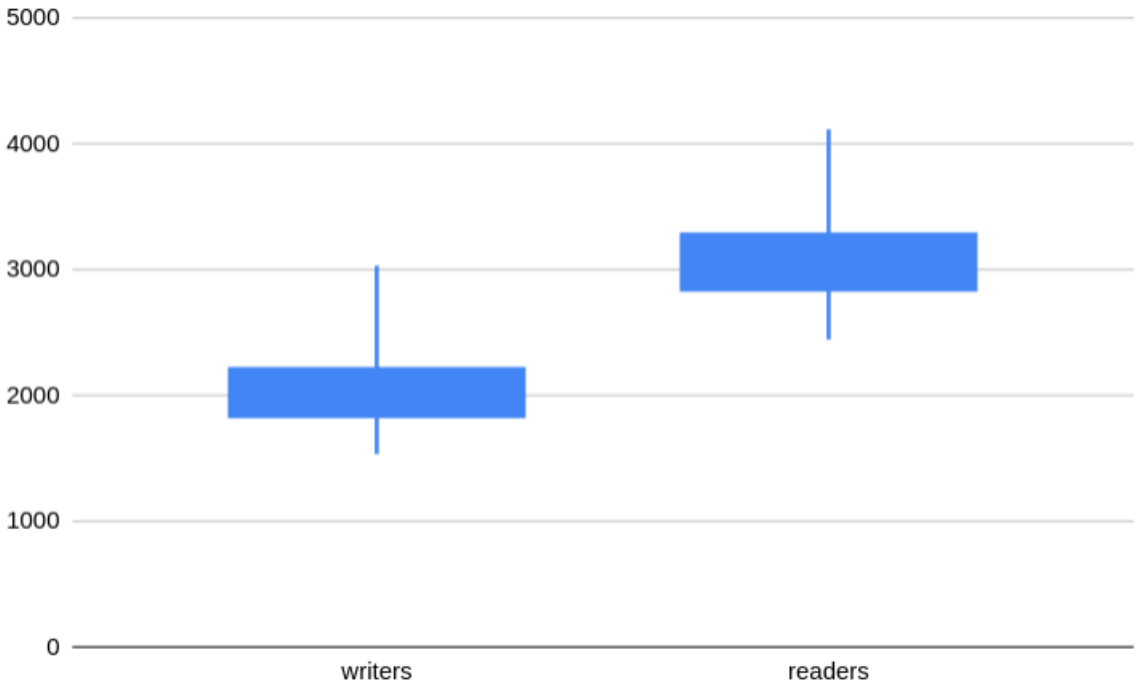
- 1 CPU
- sleeps non aléatoires dans les fonctions de lecture et d'écriture et dans les threads de lecture et d'écriture (le sleep dans la fonction de thread fait disparaître l'effet apparu lors du labo A où le premier thread travaillait plus que les autres)
- 60 threads lecteurs
- 5 threads écrivains
- valeur maximale de la donnée de 20
- écritures non aléatoires, les écrivains écrivent toujours 1 : il doit donc y avoir minimum 20 écritures et maximum 29
- scheduler round-robin (sauf pour le dernier test sans priorités)

Résultats sur 50 exécutions pour :

- 60 lecteurs avec une priorité de 10
- 5 écrivains avec une priorité de 20

moyenne d'écritures : 24
moyenne de lectures : 60

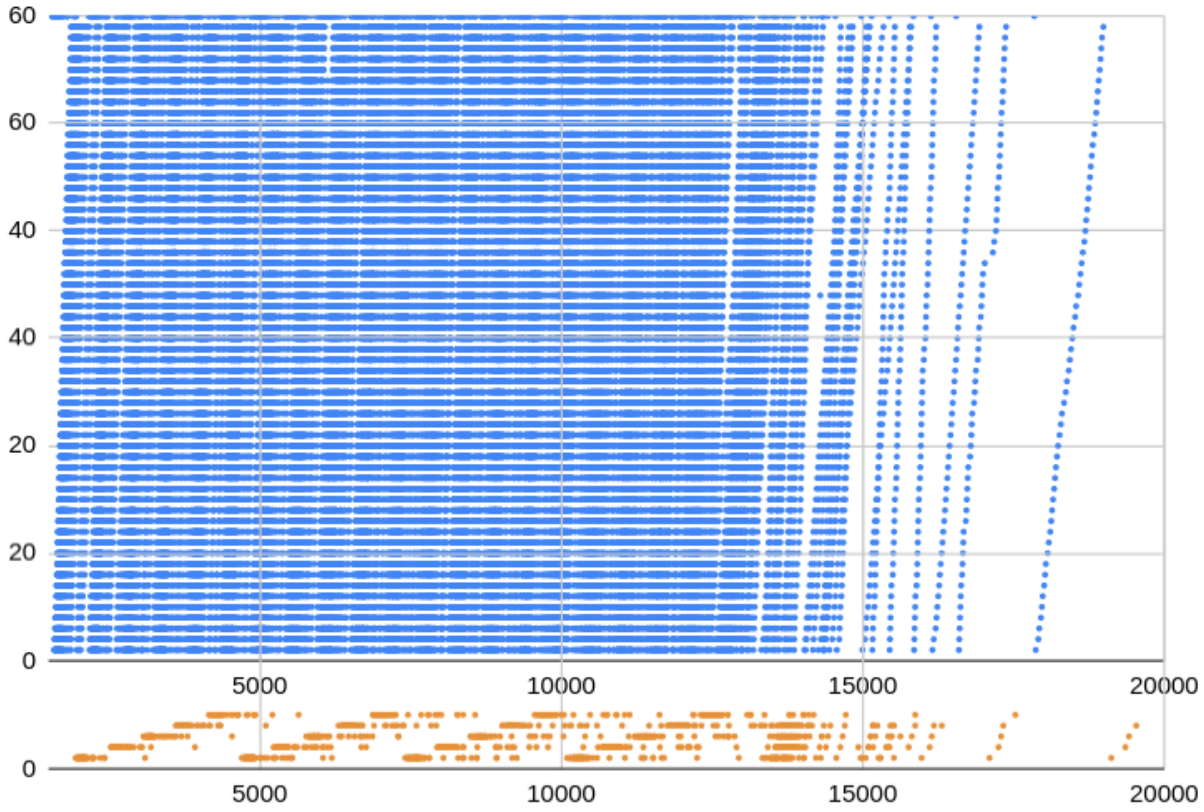


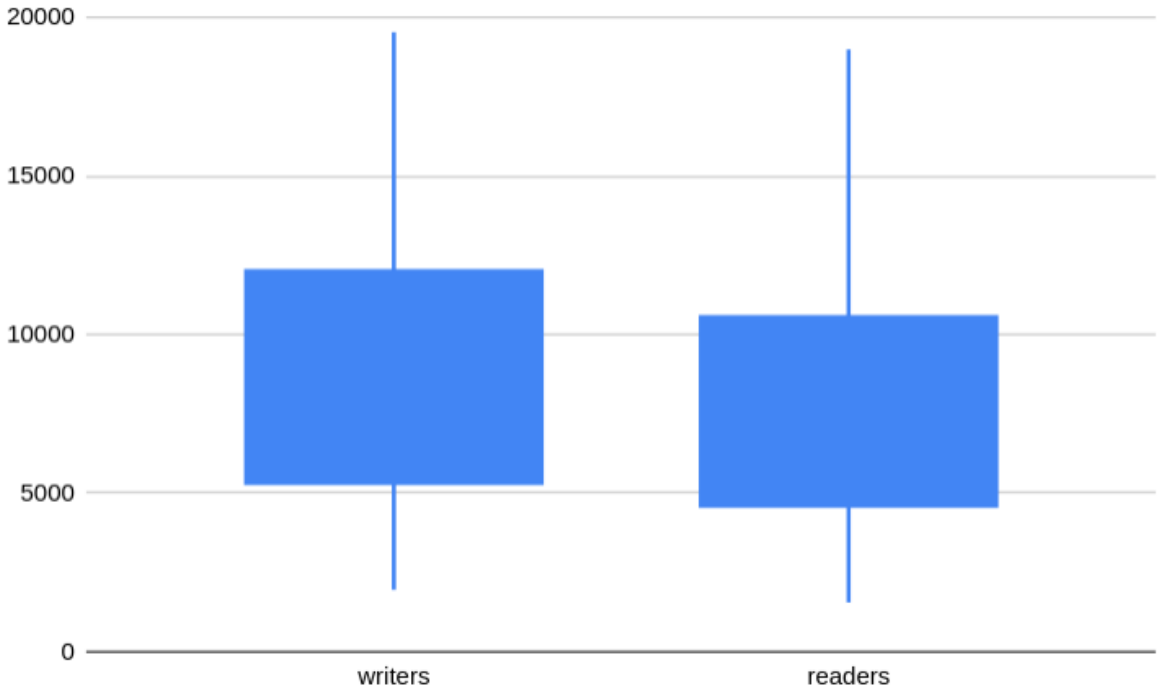


Résultats sur 50 exécutions pour :

- 60 lecteurs avec une priorité de 20
- 5 écrivains avec une priorité de 10

moyenne d'écritures : 24
moyenne de lectures : 1261.1

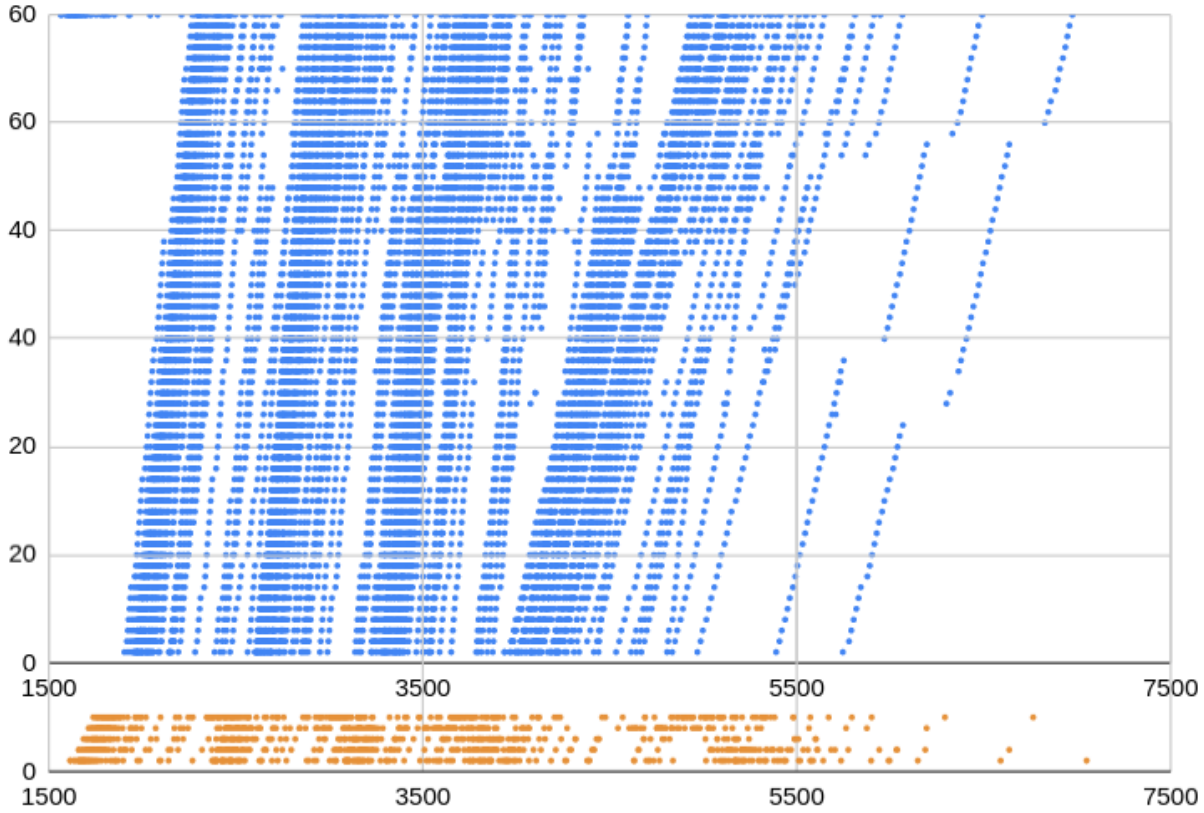


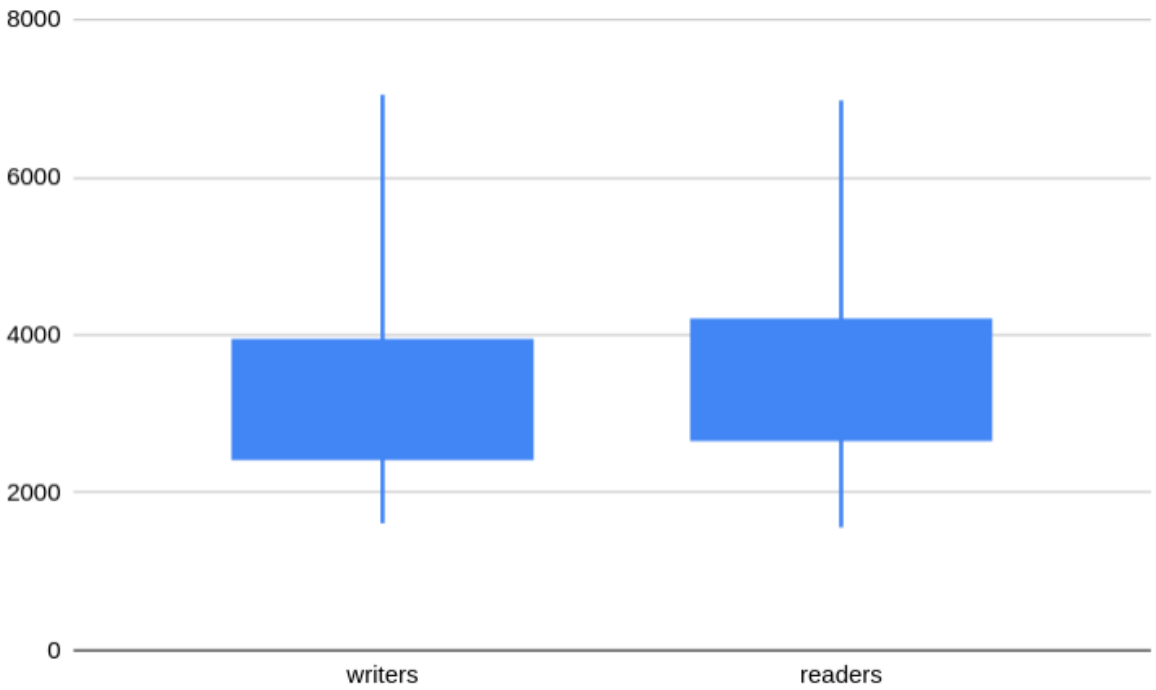


Résultats sur 50 exécutions pour :

- 60 lecteurs avec une priorité de 10
- 5 écrivains avec une priorité de 10

moyenne d'écritures : 24
moyenne de lectures : 239.18

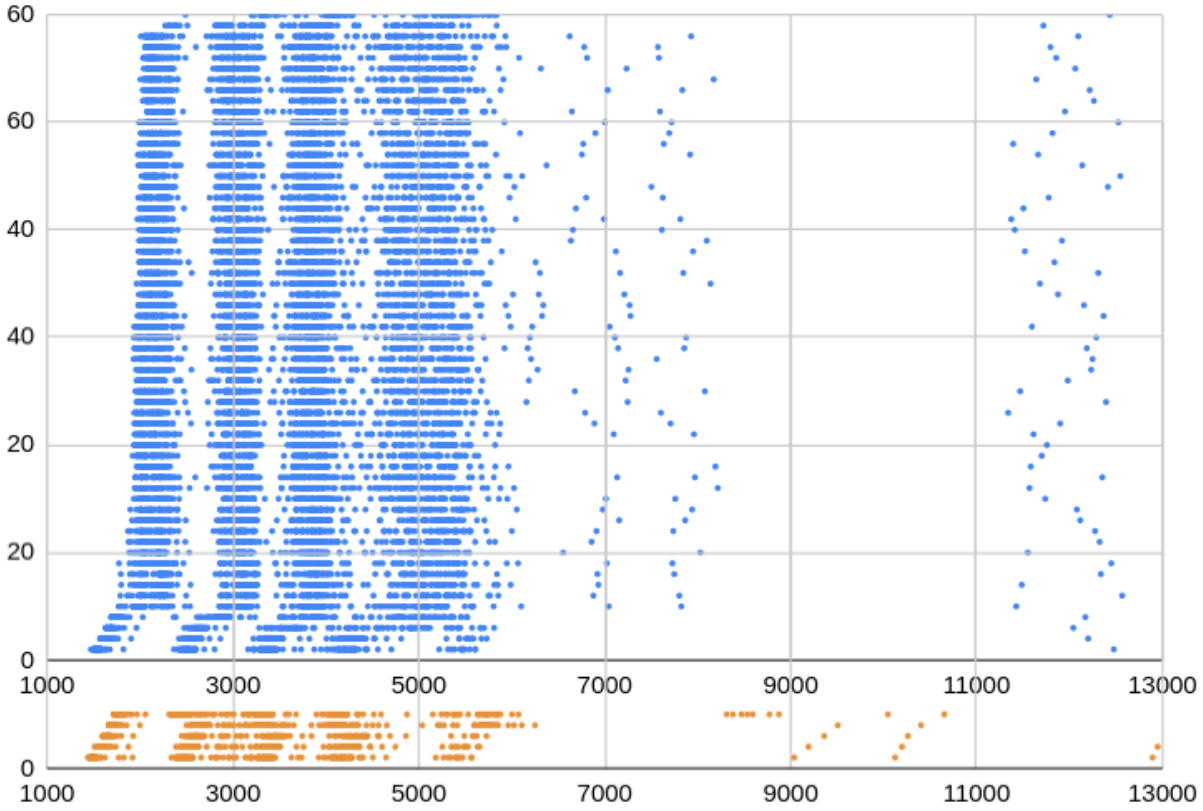


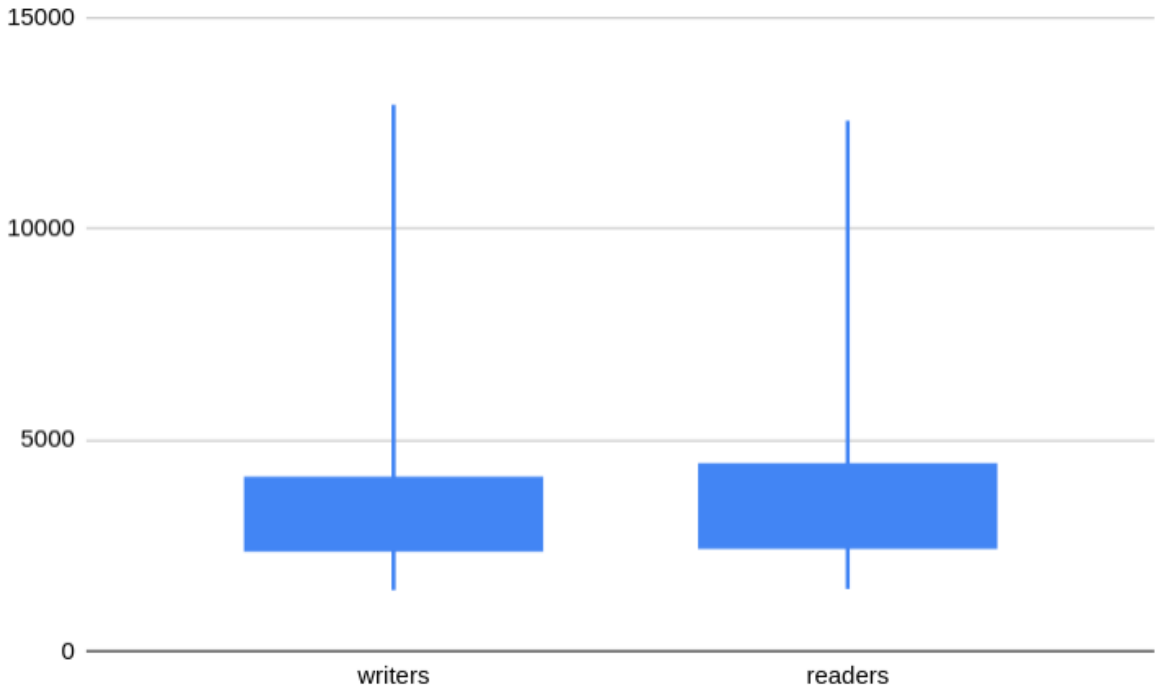


Résultats sur 50 exécutions pour :

- 60 lecteurs sans priorité
- 5 écrivains sans priorité

moyenne d'écritures : 24
moyenne de lectures : 227.54





Observations :

Le calcul des moyennes montre que si le nombre d'écritures ne change pas (il en faut toujours le même nombre avant d'arriver à la valeur maximale), le nombre de lectures est d'autant plus important que la priorité des lecteurs diminue par rapport à celle des écrivains. Comme avec les nice, le nombre de lectures moyen est réduit au minimum, i.e. au nombre de threads lecteurs, dans le cas le plus favorable aux écrivains. Contrairement au labo A, le cas où une plus grande priorité est accordée aux lecteurs diffère des cas sans priorités et à priorités égales (à voir si ce n'est pas avec l'ajout de sleeps).

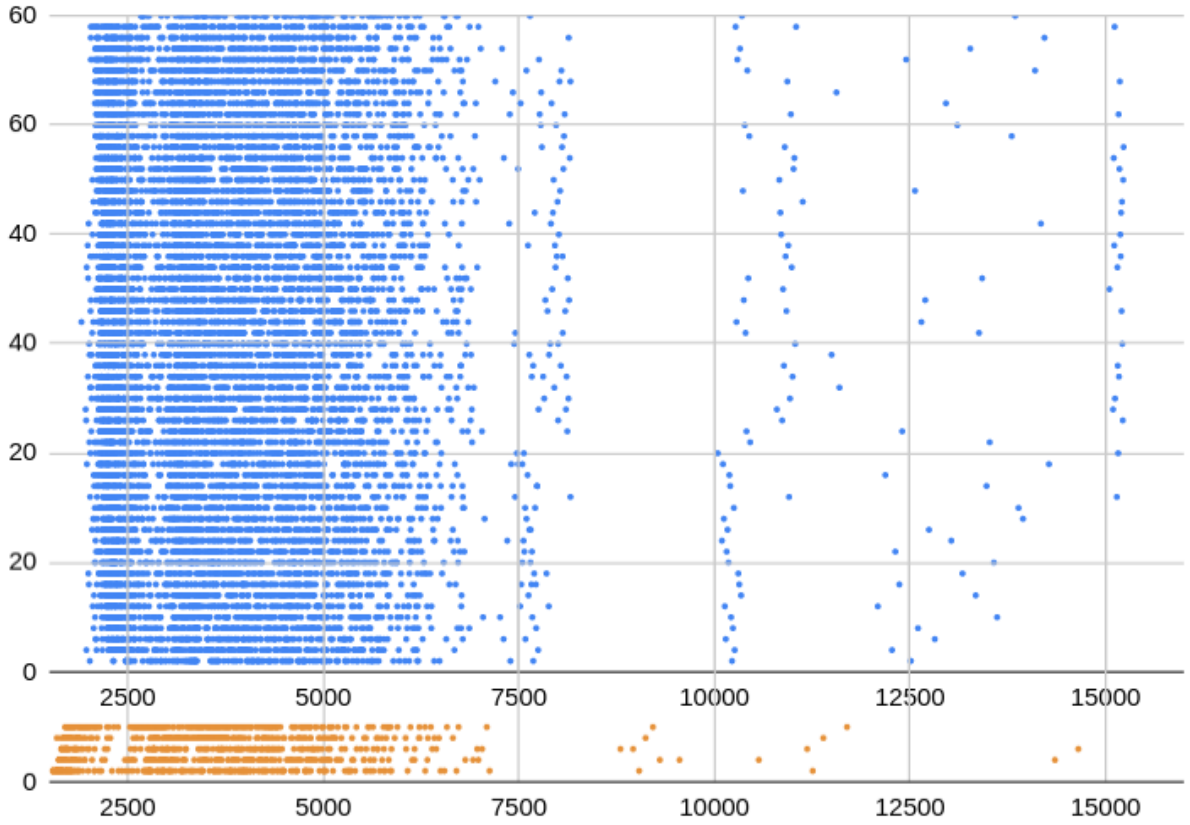
On voit aussi que le round-robin ajoute beaucoup de déterminisme (cf les deux derniers nuages de points et sortie de la commande sudo ./reader_writer 15 1 5 1 10).

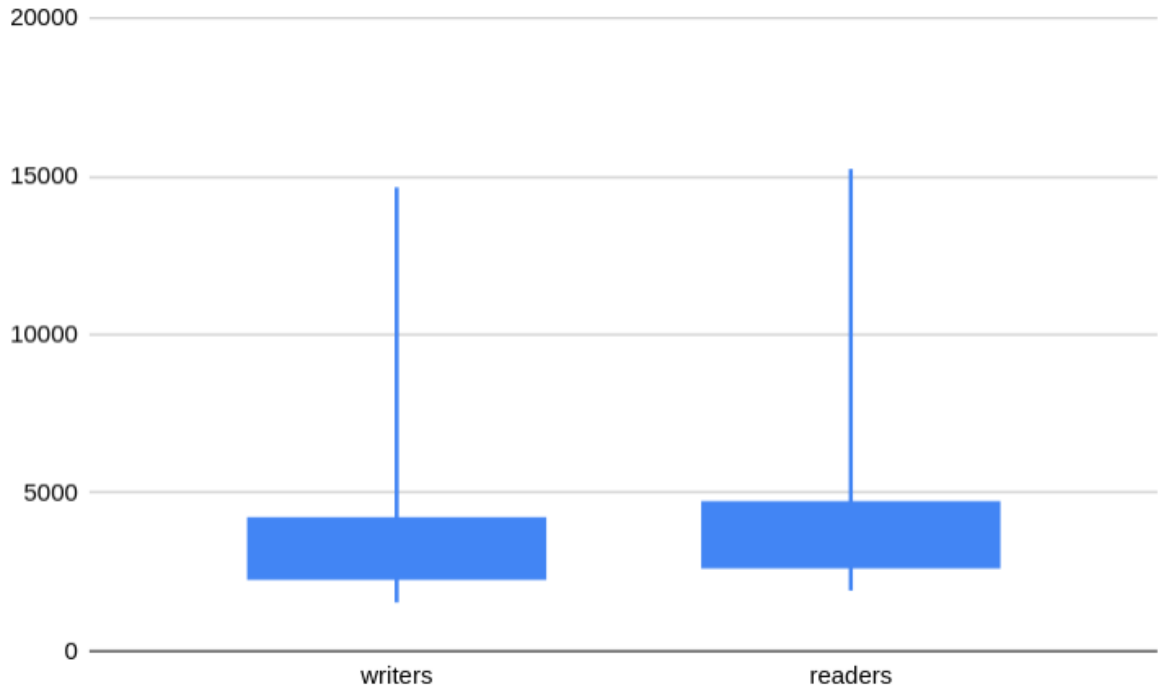
Projet lié : labb_readers_writers.zip

Tests avec les mêmes valeurs (threads, valeur max et sleeps) avec nice :
Résultats sur 50 exécutions pour :

- 60 lecteurs avec une courtoisie de 19
- 5 écrivains avec une courtoisie de -20

moyenne d'écritures : 24
moyenne de lectures : 206.3

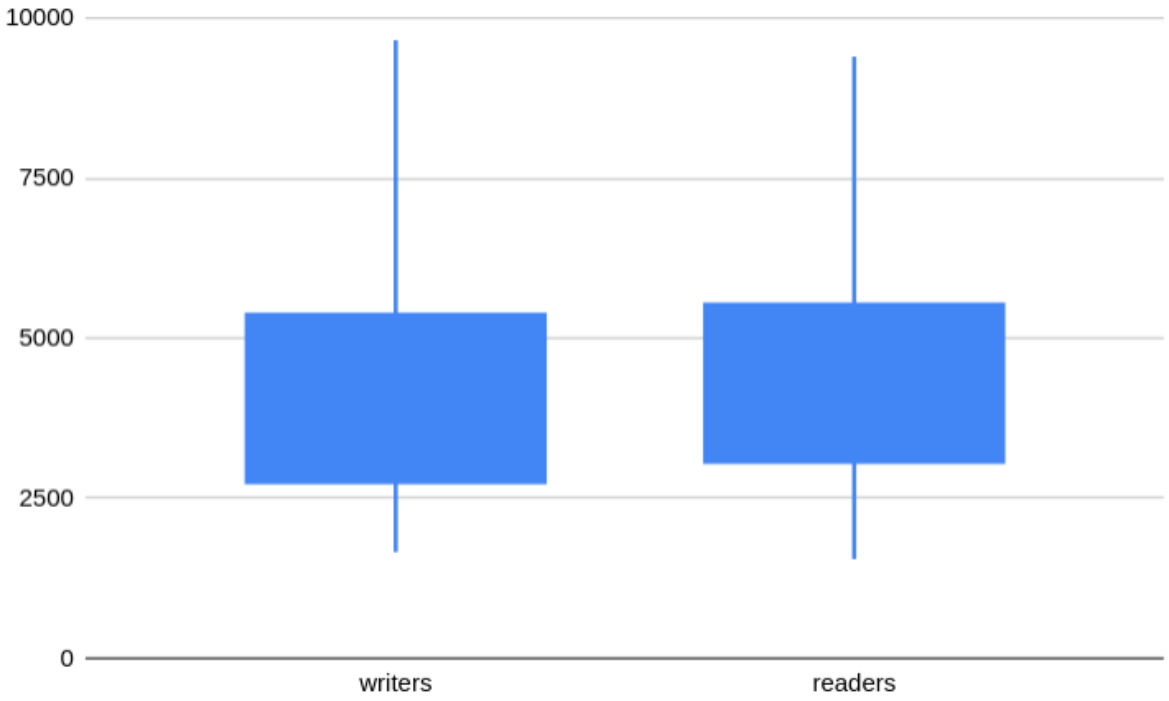




Résultats sur 50 exécutions pour :

- 60 lecteurs avec une courtoisie de -20
- 5 écrivains avec une courtoisie de 19

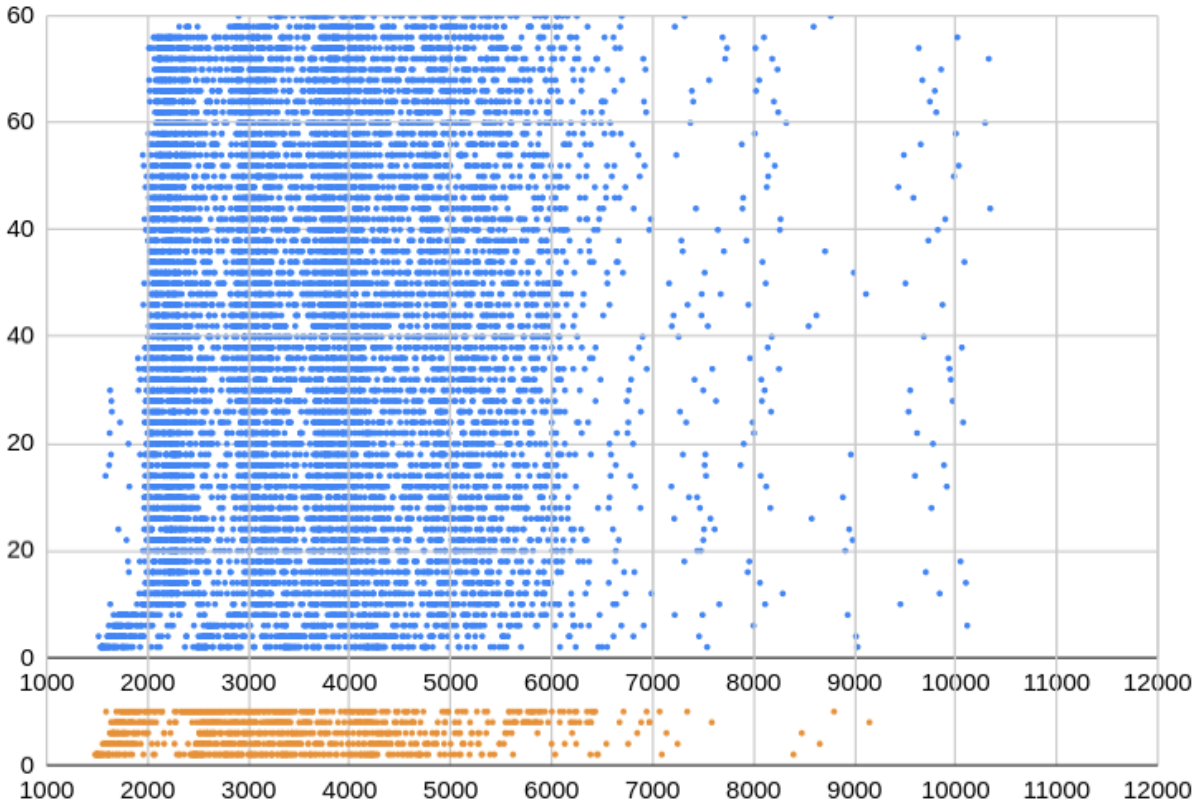
moyenne d'écritures : 24
moyenne de lectures : 322.62

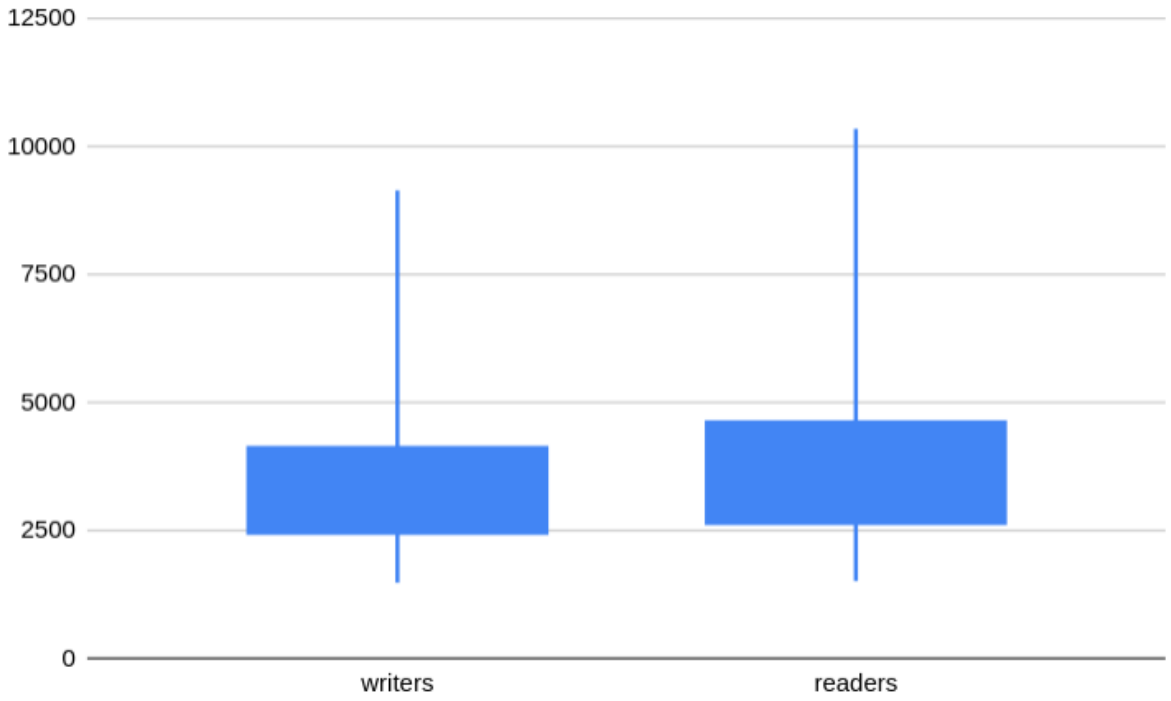


Résultats sur 50 exécutions pour :

- 60 lecteurs avec une courtoisie de 0
- 5 écrivains avec une courtoisie de 0

moyenne d'écritures : 24
moyenne de lectures : 216.96

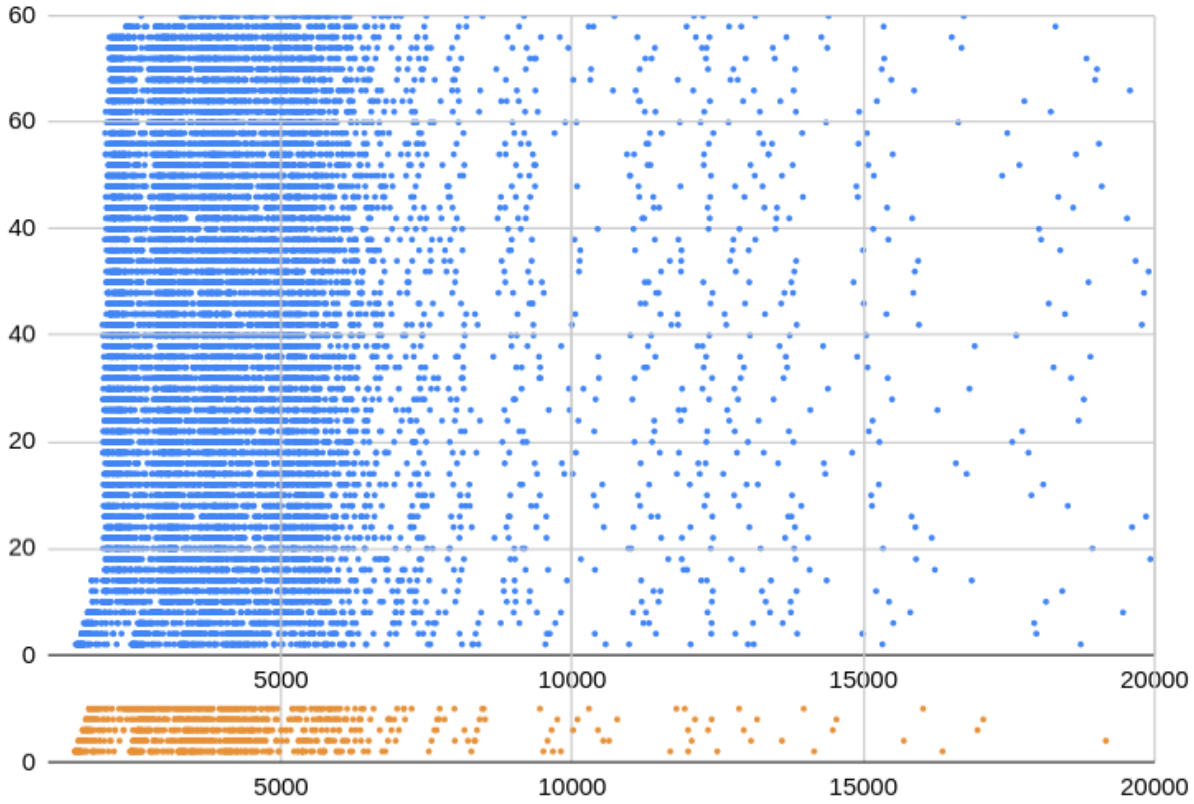


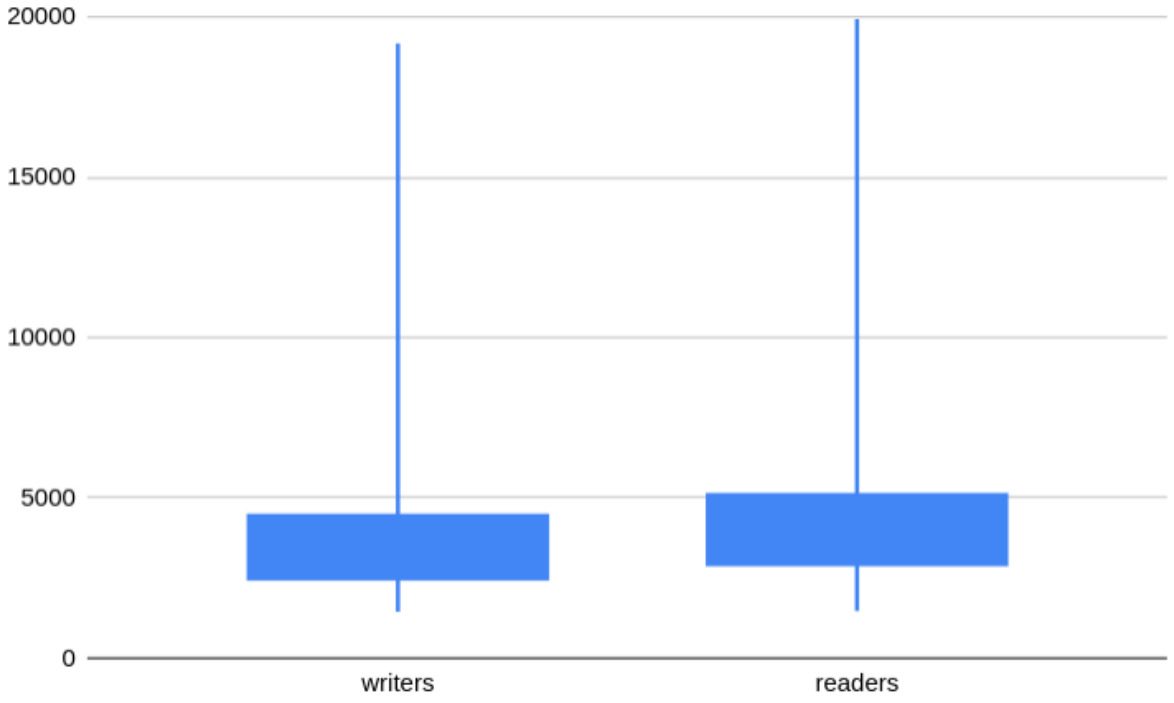


Résultats sur 50 exécutions pour :

- 60 lecteurs sans courtoisie
- 5 écrivains sans courtoisie

moyenne d'écritures : 24
moyenne de lectures : 253.32





Observations:
Les résultats sans priorité ou sans nice son bien semblables. On a aussi le même nombre moyen d'écrivains (il est donc bien déterminé par les sleep : plus ils sont courts, plus les écrivains ont le temps de revenir dans la file pour repasser dans la dernière écriture).

Avec ces valeurs de tests, le cas le plus favorable aux écrivains n'est plus le plus favorable dans l'absolu, i.e. le nombre de lectures moyennes est supérieur au nombre de threads lecteurs. Globalement, les priorités se montrent plus efficaces que les courtoisies. On le voit notamment en comparant les moyennes de lectures.

Les nouvelles valeurs de test ont permis de distinguer légèrement le cas favorable aux lecteurs des cas avec courtoisie identique et sans courtoisie (dans le bas de la courbe des lecteurs). Cette différence ne semble pourtant pas significative, surtout en comparaison des tests avec priorités.

Par rapport aux nuages de points des premiers tests, le fait que le premier thread lecteur ou écrivain travaille plus que les autres a bien été atténué.

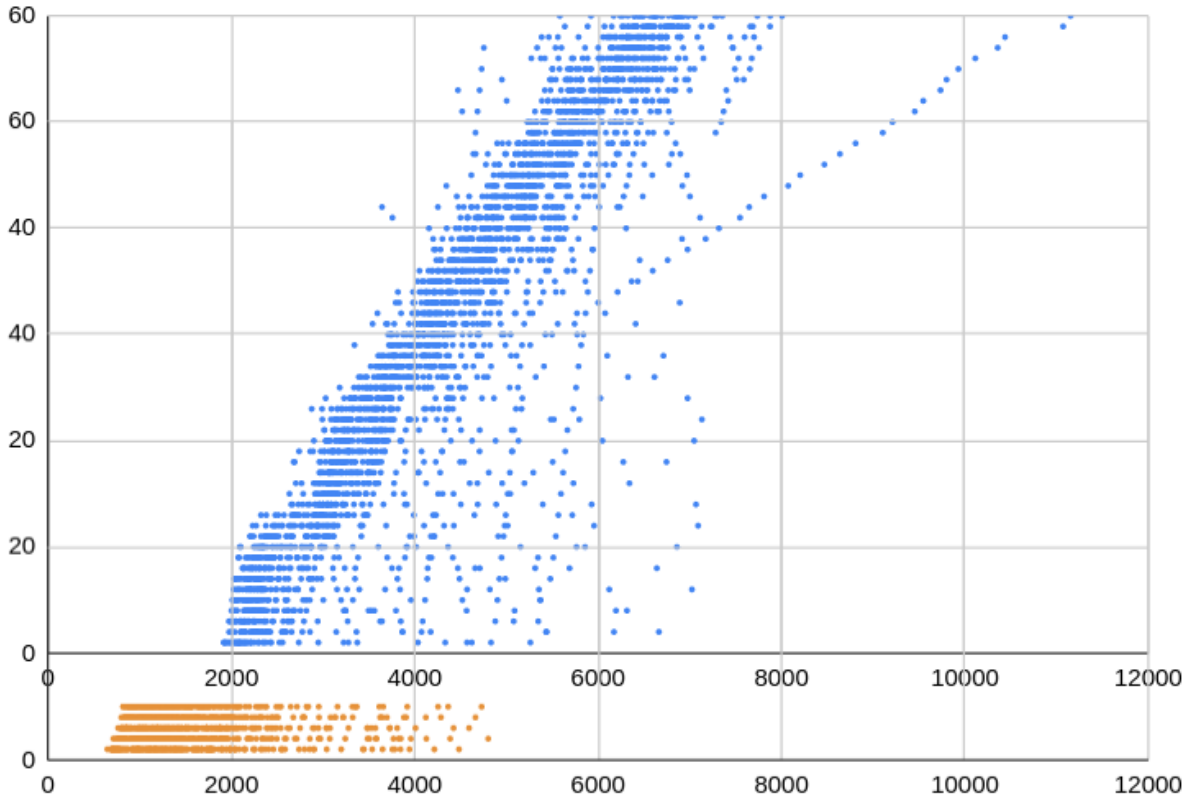
Avec les nice, les résultats n'ont été significatifs qu'avec peu de threads, peu d'écritures (avec une petite valeur maximale) alors qu'avec les priorités, ont voit des tendances de comportements se dessiner dans tous les cas.

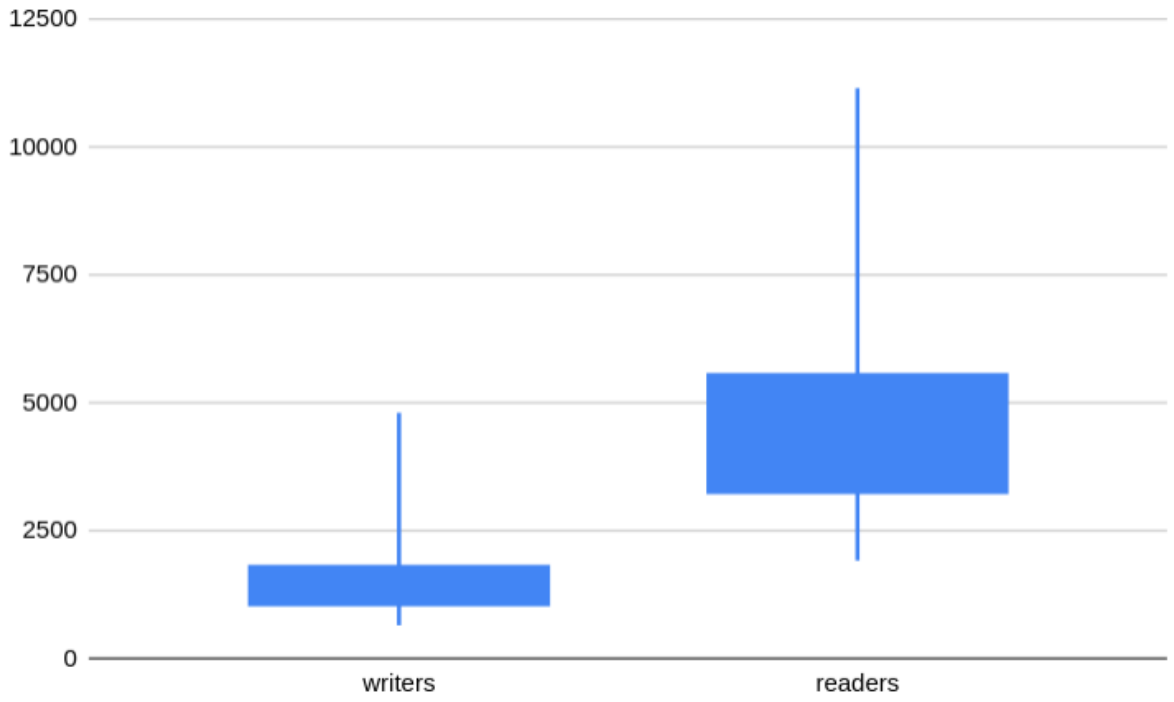
Projet lié : laba-readers-writers-v2.zip

Pour comparaison, j'ai aussi ressorti les mêmes tests avec la priorité par programmation :
Résultats sur 50 exécutions pour :

- 60 lecteurs (priorité par programmation)
- 5 écrivains (priorité par programmation)

moyenne d'écritures : 24
moyenne de lectures : 60





Observations :
Le nombre de lecteurs est de 60, c'est bien le nombre minimum qu'on peut attendre. Les graphiques ressemblent assez aux deux premiers graphiques du labo B. La différence semble se jouer sur le temps d'exécution du programme, plus long dans ce cas-ci. Mais ce temps d'exécution n'est pas fiable pour les conditions dans lesquelles j'ai fait les tests (le programme est lancé dans une VM sur ma machine où tournent beaucoup d'autres process).

Projet lié : labc_readers_writers.zip

20 mai 2021 : labo C

L'héritage de priorités ne s'applique pas à l'implémentation de priorités du labo B qui a seulement deux niveaux de priorités (il ne peut pas y avoir d'inversion de priorité). De plus, le labo B semble déjà donner les résultats attendus (résultats similaires à la priorité par programmation).

-> Est-ce que c'est quand même nécessaire d'implémenter l'héritage de priorités ? Non ce n'est pas nécessaire !

24 mai 2021 - labo D : installation de Xenomai sur Raspberry Pi 4

Pour l'installation de Xenomai, j'ai principalement suivi ce tutoriel : https://lemariva.com/en/blog/2018/07/raspberry-pi-xenomai-patching-tutorial-for-kernel-4-14-y (installation sur raspberry pi 3 de Xenomai 3 avec le kernel 4.14.36)

Complété par celui-ci : https://www.raspberrypi.org/documentation/linux/kernel/building.md

1. Upgrade de la VM

- 4Go RAM
- 4 CPU
- Enable Nested VT -x/AMD-V (virtualisation optimisée, propre au processeur AMD)

2. Téléchargements

Création des répertoires contenant le kernel linux et les fichiers compilés :
```
~/2021_TR/labd-xenomai$ mkdir rpi-kernel
~/.../rpi-kernel$ mkdir rt-kernel
```

Clone du kernel linux pour raspberry pi :
```
~/.../rpi-kernel$ git clone https://github.com/raspberrypi/linux.git
```

Clone des outils de cross-compilation (finalement pas utile) :
```
~/.../rpi-kernel$ git clone https://github.com/raspberrypi/tools.git --depth 3
```

Téléchargement de la dernière version stable de Xenomai :
```
~/.../rpi-kernel$ wget https://xenomai.org/downloads/xenomai/stable/xenomai-3.1.tar.bz2
~/.../rpi-kernel$ tar xf xenomai-3.1.tar.bz2
```

Téléchargement du patch ipipe :
```
~/.../rpi-kernel$ wget
https://xenomai.org/downloads/ipipe/v4.x/arm/ipipe-core-4.19.128-arm-9.patch
```

J'ai pris la version 4.19.128 du patch car c'est la dernière version la plus proche d'un kernel linux de raspberry pi (https://github.com/raspberrypi/linux/commits/rpi-4.19.y).

Checkout sur la branche de la version 4.19 du kernel :
```
~/.../rpi-kernel/linux$ git checkout rpi-4.19.y
```

Ajout d'un repository remote pour pouvoir accéder à la version 4.19.128 car raspberry s'arrête à la version 127 :
```
~/.../rpi-kernel/linux$ git remote add stable
https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git
```

Téléchargement des informations sur le repository :
```
~/.../rpi-kernel/linux$ git fetch stable
```

Merge du repository raspberry avec le nouveau :
```
~/.../rpi-kernel/linux$ git merge v4.19.128
```

-> pas de conflit \o/

Je ne l'ai pas fait ici mais ça aurait été le bon moment de télécharger les outils nécessaires pour la compilation du kernel (selon le second tutoriel) :
```
sudo apt install git bc bison flex libssl-dev make
```

3. Patch du kernel

Application du patch :
```
~/.../rpi-kernel$ xenomai-3.1/scripts/prepare-kernel.sh --linux=linux/ --arch=arm
--ipipe=ipipe-core-4.19.128-arm-9.patch --verbose
```

Erreur :
```
[...]
prepare-kernel.sh: Unable to patch kernel 4.19.128 with ipipe-core-4.19.128-arm-9.patch.
```

sur les fichiers :
```
checking file drivers/irqchip/irq-bcm2835.c
checking file drivers/irqchip/irq-bcm2836.c
```

Il s'agit des processeurs utilisés dans les raspberry 1 et 2 (documentation). Puisqu'ils ne sont pas utiles pour le raspberry 4, je supprime les lignes 8938 à 9088 qui font référence aux deux fichiers problématiques, puis je relance la même commande :
```
~/.../rpi-kernel$ xenomai-3.1/scripts/prepare-kernel.sh --linux=linux/ --arch=arm
--ipipe=ipipe-core-4.19.128-arm-9.patch --verbose
```

Et ça fonctionne \o/ :
```
[...]
I-pipe core/arm #9 installed.
Links installed.
Build system ready.
```

4. Configuration (pour compilation en 32 bits)

Configuration des variables suivantes selon le premier tutoriel :
```
export ARCH=arm
export
CROSS_COMPILE=~/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabi
hf-raspbian/bin/arm-linux-gnueabihf-
export INSTALL_MOD_PATH=~/2021_TR/labd-xenomai/rpi-kernel/rt-kernel
export INSTALL_DTBS_PATH=~/2021_TR/labd-xenomai/rpi-kernel/rt-kernel
```

Et de la variable suivante suivant le second tutoriel (propre au raspberry pi 4 : armv7l) :
```
export KERNEL=kernel7l
```

Application de la configuration (second tutoriel) :

```
~/.../rpi-kernel/linux$ make bcm2711_defconfig
```

Ici j'ai eu deux erreurs car je n'avais pas installé les outils recommandés (bison et flex).
Première erreur :

```
/bin/sh: 1:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
  HOSTCC  scripts/basic/fixdep
/bin/sh: 1:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
  HOSTCC  scripts/kconfig/conf.o
  YACC  scripts/kconfig/zconf.tab.c
/bin/sh: 1: bison: not found
scripts/Makefile.lib:196: recipe for target 'scripts/kconfig/zconf.tab.c' failed
make[1]: *** [scripts/kconfig/zconf.tab.c] Error 127
Makefile:534: recipe for target 'bcm2711_defconfig' failed
make: *** [bcm2711_defconfig] Error 2


-> sudo apt install bison
-> sudo apt install flex
```

Puis j'ai l'erreur suivante qui me dit que le fichier arm-linux-gnueabihf-gcc n'a pas été trouvé:

```
/bin/sh: 1:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
/bin/sh: 1:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
  LEX   scripts/kconfig/zconf.lex.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTLD  scripts/kconfig/conf
./scripts/gcc-version.sh: 26: ./scripts/gcc-version.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
./scripts/gcc-version.sh: 27: ./scripts/gcc-version.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
./scripts/gcc-version.sh: 29: ./scripts/gcc-version.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
./scripts/gcc-version.sh: 26: ./scripts/gcc-version.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
./scripts/gcc-version.sh: 27: ./scripts/gcc-version.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
./scripts/gcc-version.sh: 29: ./scripts/gcc-version.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
init/Kconfig:17: syntax error
init/Kconfig:16: invalid option
./scripts/clang-version.sh: 15: ./scripts/clang-version.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
./scripts/gcc-plugin.sh: 11: ./scripts/gcc-plugin.sh:
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-ra
spbian/bin/arm-linux-gnueabihf-gcc: not found
scripts/kconfig/Makefile:104: recipe for target 'bcm2711_defconfig' failed
make[1]: *** [bcm2711_defconfig] Error 1
Makefile:534: recipe for target 'bcm2711_defconfig' failed
```

```
make: *** [bcm2711_defconfig] Error 2
```

J'affiche les Informations sur le fichier :

```
file ${CROSS_COMPILE}gcc-4.8.3
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian/bin/arm-linux-gnueabihf-gcc-4.8.3: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.15, stripped
```

On voit que gcc-4.8.3 est en 32 bits mais ma VM est en 64 bits, je n'ai pas les librairies pour l'exécuter. J'abandonne la version précompilée fournie par raspberry et je prend la version fournie par ubuntu (selon ce tutoriel sur la cross compilation pour raspberry pi en C++).

Installation du compilateur C pour processeur ARM :

```
sudo apt install -y gcc-arm-linux-gnueabihf
```

Configuration de la variable CROSS_COMPILE avec le nouveau compilateur :

```
export CROSS_COMPILE=arm-linux-gnueabihf-
```

Je peux maintenant appliquer la configuration :

```
~/.../rpi-kernel/linux$ make bcm2711_defconfig
#
# configuration written to .config
#
```

Configuration d'options (premier tutoriel) :

```
~/.../rpi-kernel/linux$ make menuconfig
```

Erreur:

```
*
* Unable to find the ncurses package.
* Install ncurses (ncurses-devel or libncurses-dev
* depending on your distribution).
*
scripts/kconfig/Makefile:228: recipe for target 'scripts/kconfig/.mconf-cfg' failed
make[1]: *** [scripts/kconfig/.mconf-cfg] Error 1
Makefile:534: recipe for target 'menuconfig' failed
make: *** [menuconfig] Error 2

-> sudo apt install libncurses-dev
```

Quand je relance `make menuconfig`, une magnifique interface graphique apparaît et je peux configurer les options suivantes selon le premier tutoriel :

- Disable CPU Frequency scaling: CPU Power Management → CPU Frequency scaling → CPU Frequency scaling
- Disable KGDB: kernel debugger: KGDB: kernel debugger → Kernel Hacking

Par contre je n'ai pas trouvé les options suivantes :

- Disable Allow for memory compaction: Kernel Features → Allow for memory compaction
- Disable Contiguous Memory Allocator: Kernel Features → Contiguous Memory Allocator

Attention : il aurait aussi fallu renommer la version (General setup -> Local version), cf. problème posé lors de l'installation du kernel sans patch

5. Compilation (temps estimé : un demi siècle - temps réel : moins d'une heure)

Installation des outils que j'aurai dû installer au début (second tutoriel) :

```
sudo apt install libssl-dev
```

Compilation du kernel (-j4 car j'ai 4 CPU) :

```
~/.../rpi-kernel/linux$ make -j4 zImage modules dtbs
~/.../rpi-kernel/linux$ make -j4 modules_install dtbs_install
```

sortie :

```
[...]
DEPMOD  4.19.128-v7l+
```

Création de l'image :

```
~/.../rpi-kernel/linux$  mkdir $INSTALL_MOD_PATH/boot
~/.../rpi-kernel/linux$  ./scripts/mkknlimg ./arch/arm/boot/zImage
$INSTALL_MOD_PATH/boot/$KERNEL.img
```

sortie :

```
Version: Linux version 4.19.128-v7l+ (lbin@tr-ubuntu18-server) (gcc version 7.5.0
(Ubuntu/Linaro 7.5.0-3ubuntu1~18.04)) #1 SMP Mon May 24 18:22:00 UTC 2021
DT: y
DDT: y
270x: y
283x: y
```

Copie de l'image sur le raspberry :

```
~/.../rpi-kernel/rt-kernel$ tar czf ../xenomai-kernel.tgz *
~/.../rpi-kernel$ scp xenomai-kernel.tgz lbin@192.168.188.47:
```

6. Installation de l'image sur le raspberry

Extraction de l'image dans le répertoire xenomai :

```
~ $ mkdir xenomai
~/xenomai $ tar xf ../xenomai-kernel.tgz
```

Copie des fichiers aux bons endroits (en suivant le premier tutoriel) :

```
~/xenomai $ sudo cp *.dtb /boot/
~/xenomai/boot $ sudo cp -rd kernel7l.img /boot/
~/xenomai/lib $ rm modules/4.19.128-v7l+/source
```

(suppression du lien symbolique obsolète)

```
~/xenomai/lib $ sudo cp -dr * /lib/
~/xenomai/overlays $ sudo cp -d * /boot/overlays/
```

Edition du fichier /boot/config.txt :

```
~/xenomai $ sudo vi /boot/config.txt
```

ajout de :

```
kernel=${zImage name}
device_tree=bcm2711-rpi-4-b.dtb
```

Edition du fichier /boot/cmdline.txt :

```
~/xenomai $ sudo vi /boot/cmdline.txt
```

ajout de :

```
dwc_otg.fiq_enable=0 dwc_otg.fiq_fsm_enable=0 dwc_otg.nak_holdoff=0
```

7. Redémarrage

```
sudo reboot
ssh lbin@192.168.188.47
lbin@raspberrypi:~ $ uname -a
Linux raspberrypi 4.19.128-v7l+ #1 SMP Mon May 24 18:22:00 UTC 2021 armv7l GNU/Linux
```

-> tout est OK \o\ /o/ <o/ \o> \o/

27 mai 2021 - labo D : installation de Xenomai

8. Compilation des outils Xenomai

Téléchargement de la dernière version stable de Xenomai sur le Raspberry :

```
lbin@raspberrypi:~ $ wget https://xenomai.org/downloads/xenomai/stable/xenomai-3.1.tar.bz2
lbin@raspberrypi:~ $ tar xf xenomai-3.1.tar.bz2
```

Installation des outils de compilation :

```
lbin@raspberrypi:~ $ sudo apt install automake
lbin@raspberrypi:~ $ sudo apt install libtool
```

Compilation des outils Xenomai

```
lbin@raspberrypi:~/xenomai-3.1 $ ./scripts/bootstrap --with-core=cobalt -enable-debug=partial
lbin@raspberrypi:~/xenomai-3.1 $ ./configure --with-core=cobalt --enable-smp
lbin@raspberrypi:~/xenomai-3.1 $ make -j4
lbin@raspberrypi:~/xenomai-3.1 $ sudo make install -j4
```

Lancement de la commande latency :

```
lbin@raspberrypi:~/xenomai-3.1 $ sudo /usr/xenomai/bin/latency
sudo /usr/xenomai/bin/latency
== Sampling period: 1000 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:01  (periodic user-mode task, 1000 us period, priority 99)
RTH|----lat min|----lat avg|----lat max|-overrun|---msw|---lat best|--lat worst
RTD|      0.685|      3.109|     18.759|       0|     0|      0.685|     18.759
RTD|      0.610|      2.218|     13.554|       0|     0|      0.610|     18.759
RTD|      0.665|      1.242|     12.183|       0|     0|      0.610|     18.759
RTD|      0.701|      3.030|     18.516|       0|     0|      0.610|     18.759
RTD|      0.756|      2.024|     13.701|       0|     0|      0.610|     18.759
RTD|      0.126|      1.484|     13.200|       0|     0|      0.126|     18.759
RTD|     -0.726|      2.887|     20.366|       0|     0|     -0.726|     20.366
RTD|      0.106|      0.702|      9.866|       0|     0|     -0.726|     20.366
RTD|      0.143|      1.576|     16.680|       0|     0|     -0.726|     20.366
RTD|      0.031|      1.875|     19.476|       0|     0|     -0.726|     20.366
RTD|      0.031|      0.551|      6.642|       0|     0|     -0.726|     20.366
RTD|      0.122|      0.430|      6.308|       0|     0|     -0.726|     20.366
RTD|      0.085|      0.546|      5.511|       0|     0|     -0.726|     20.366
RTD|     -0.009|      0.508|      4.918|       0|     0|     -0.726|     20.366
RTD|      0.158|      0.580|      5.121|       0|     0|     -0.726|     20.366
RTD|      0.250|      0.624|      5.472|       0|     0|     -0.726|     20.366
RTD|      0.212|      0.718|      5.786|       0|     0|     -0.726|     20.366
RTD|      0.193|      0.555|      5.323|       0|     0|     -0.726|     20.366
RTD|      0.081|      0.634|     11.951|       0|     0|     -0.726|     20.366
RTD|      0.006|      0.519|      7.895|       0|     0|     -0.726|     20.366
RTD|      0.025|      1.533|     17.580|       0|     0|     -0.726|     20.366
```

Sauvegarde de la carte SD sur laquelle a été installé Xenomai

Affichage des informations sur les périphériques :

```
[lbin@nibbler ~]$ lsblk
NAME                                    MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
mmcblk0                                 179:0   0      15G  0 disk
├─mmcblk0p1                             179:1   0   256M  0 part  /run/media/lbin/boot
└─mmcblk0p2                             179:2   0  14.8G  0 part  /run/media/lbin/rootfs
[...]
```

Vérification de la présence de la carte SD et de ses partitions dans /dev :

```
[lbin@nibbler ~]$ ls -l /dev
[...]
brw-rw----.  1 root disk      179,     0 May 27 12:44 mmcblk0
brw-rw----.  1 root disk      179,     1 May 27 12:44 mmcblk0p1
brw-rw----.  1 root disk      179,     2 May 27 12:44 mmcblk0p2
[...]
```

Démontage des partitions de la carte SD pour pouvoir la copier :

```
[lbin@nibbler ~]$ umount /dev/mmcblk0p1
[lbin@nibbler ~]$ umount /dev/mmcblk0p2
```

Les partitions ont bien été démontées :

```
[lbin@nibbler ~]$ lsblk
NAME                                    MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
mmcblk0                                 179:0    0    15G  0 disk
├─mmcblk0p1                             179:1    0   256M  0 part
└─mmcblk0p2                             179:2    0  14.8G  0 part
[...]
```

Copie de la carte avec dd et compression avec xz :

```
[lbin@nibbler ~]$ sudo dd if=/dev/mmcblk0 of=/dev/stdout bs=1024M | xz > xenomai_backup.img.xz
```

Selon la carte SD, ça peut prendre du temps (une heure pour moi). On peut vérifier l'avancée de la compression avec pkill lancé depuis un autre terminal :

```
[lbin@nibbler ~]$ pkill -USR1 xz
```

Qui affiche les informations suivantes dans le premier terminal :

```
  --- %   1,019.9 MiB / 6,144.0 MiB = 0.166   3.2 MiB/s      32:18
```

Résultat de la commande complétée :

```
[lbin@nibbler ~]$ sudo dd if=/dev/mmcblk0 of=/dev/stdout bs=1024M | xz > xenomai_backup.img.xz
[sudo] password for lbin:
  --- %   1,019.9 MiB / 6,144.0 MiB = 0.166   3.2 MiB/s      32:18
  --- %      2,325.9 MiB / 14.4 GiB = 0.157   4.4 MiB/s      56:15
15+0 records in
15+0 records out
16106127360 bytes (16 GB, 15 GiB) copied, 3643.17 s, 4.4 MB/s
```

27 mai 2021 - labo D : installation de Preempt-RT sur Raspberry Pi 4

1. Préparation

La version 4.19 du kernel est déjà par raspberry déjà patchée avec Preempt-RT (branche rpi-4.19.y-rt) mais elle est basée sur la version 4.19.71 (cf historique des commits), trop éloignée de la version 4.19.128 utilisée précédemment.

Je repars donc du repository du projet précédent et pour éviter de re télécharger tous les fichiers, je crée un nouvel espace de travail lié au repository existant :

```
~/2021_TR/labd-xenomai/rpi-kernel/linux$ git worktree add -b rpi-4.19.y-preempt-rt
../../../labd-preempt-rt/linux rpi-4.19.y
```

Sortie :

```
Preparing ../../../labd-preempt-rt/linux (identifier linux)
Checking out files: 100% (62377/62377), done.
HEAD is now at 1e5eb735d807 Merge tag 'v4.19.128' into rpi-4.19.y
```

J'ai donc créé la branche rpi-4.19.y-preempt-rt dont les fichiers se retrouvent dans le répertoire labd-preempt-rt/linux.

2. Application du patch

Téléchargement du patch Preempt-RT :

```
~/2021_TR/labd-preempt-rt$ wget
http://cdn.kernel.org/pub/linux/kernel/projects/rt/4.19/older/patch-4.19.127-rt55.patch.xz
~/2021_TR/labd-preempt-rt$ unxz patch-4.19.127-rt55.patch.xz
```

Test de la commande patch avec l'option dry-run :

```
~/2021_TR/labd-preempt-rt/linux$ patch -p1 --dry-run < ../patch-4.19.127-rt55.patch
```

Pas d'erreur :

```
~/2021_TR/labd-preempt-rt/linux$ echo $?
0
```

Application du patch :

```
~/2021_TR/labd-preempt-rt/linux$ patch -p1 < ../patch-4.19.127-rt55.patch
```

3. Configuration

Suivant ce tutoriel :
https://lemariva.com/blog/2019/09/raspberry-pi-4b-preempt-rt-kernel-419y-performance-test

Création du répertoire contenant les fichiers compilés :

```
~/2021_TR/labd-preempt-rt$ mkdir rt-kernel
```

Configuration des variables :

```
export ARCH=arm
export INSTALL_MOD_PATH=~/2021_TR/labd-preempt-rt/rt-kernel
export INSTALL_DTBS_PATH=~/2021_TR/labd-preempt-rt/rt-kernel
```

Pour l'outil de cross-compilation, je ne suis pas le tutoriel mais je reprends l'outil utilisé pour l'installation de Xenomai :

```
export CROSS_COMPILE=arm-linux-gnueabihf-
```

Configuration de la version ARM propre au Raspberry Pi 4:

```
export KERNEL=kernel7l
```

Application de la configuration :

```
~/2021_TR/labd-preempt-rt/linux$ make bcm2711_defconfig
```

4. Compilation du kernel

```
~/2021_TR/labd-preempt-rt/linux$ make -j4 zImage modules dtbs
~/2021_TR/labd-preempt-rt/linux$ make -j4 modules_install dtbs_install
```

Sortie :

```
[...]
DEPMOD  4.19.128-rt55-v7l+
```

Création de l'image du kernel :

```
~/2021_TR/labd-preempt-rt/linux$ mkdir $INSTALL_MOD_PATH/boot
~/2021_TR/labd-preempt-rt/linux$ ./scripts/mkknlimg ./arch/arm/boot/zImage
$INSTALL_MOD_PATH/boot/$KERNEL.img
```

Sortie :

```
Version: Linux version 4.19.128-rt55-v7l+ (lbin@tr-ubuntu18-server) (gcc version 7.5.0
(Ubuntu/Linaro 7.5.0-3ubuntu1~18.04)) #1 SMP Thu May 27 15:18:00 UTC 2021
DT: y
DDT: y
270x: y
283x: y
```

Changement du nom de l'image :

```
~/2021_TR/labd-preempt-rt/rt-kernel/boot$ mv $KERNEL.img kernel7l_rt.img
```

Compression et copie de l'image sur le raspberry :

```
~/2021_TR/labd-preempt-rt/rt-kernel$ tar czf ../rt-kernel.tgz *
~/2021_TR/labd-preempt-rt$ scp rt-kernel.tgz lbin@192.168.188.47:
```

5. Installation de l'image sur le raspberry

Extraction de l'image dans le répertoire rt-kernel :

```
lbin@raspberrypi:~ $ mkdir rt-kernel && cd rt-kernel
lbin@raspberrypi:~/rt-kernel $ tar xf ../rt-kernel.tgz
```

Pour pouvoir installer les deux images sur la même carte SD, je vérifie les différences entre les fichiers :

- les images ont des noms différents puisque j'ai renommé l'image avec patch preempt-rt en kernel7l_rt.img
- les .dtb sont identiques :

```
lbin@raspberrypi:~ $ diff xenomai/bcm2711-rpi-4-b.dtb rt-kernel/bcm2711-rpi-4-b.dtb
```

- Les fichiers du sous-répertoire overlays sont identiques :

```
lbin@raspberrypi:~/xenomai/overlays $ for file in *; do diff "$file" "../../rt-kernel/overlays/$file"; done
```

- Les fichiers du dossier /lib sont différents mais sont placés les répertoires sont nommés différemment :

```
lbin@raspberrypi:~/xenomai $ for file in */**; do diff "$file" "../rt-kernel/$file"; done
lbin@raspberrypi:~/rt-kernel $ for file in */**; do diff "$file" "../xenomai/${file/-rt55/}"; done
lbin@raspberrypi:~ $ ls xenomai/lib/modules/ && ls rt-kernel/lib/modules/
4.19.128-v7l+
4.19.128-rt55-v7l+
```

Pour installer la nouvelle image, je dois donc uniquement copier l'image et le contenu du dossier /lib :

```
lbin@raspberrypi:~ $ sudo cp rt-kernel/boot/kernel7l_rt.img /boot/
lbin@raspberrypi:~ $ sudo cp -rd rt-kernel/lib/* /lib/
```

Edition du fichier /boot/config.txt :

```
sudo vi /boot/config.txt
```

Pour commenter la ligne qui lance l'image par défaut (contenant ici Xenomai) pour la remplacer par la nouvelle image :

```
#kernel=${zImage name}
kernel=kernel7l_rt.img
```

Edition du fichier /boot/cmdline.txt :

```
~/xenomai $ sudo vi /boot/cmdline.txt
```

suppression de :

```
dwc_otg.fiq_enable=0 dwc_otg.fiq_fsm_enable=0 dwc_otg.nak_holdoff=0
```

6. Redémarrage

```
sudo reboot
ssh lbin@192.168.188.47
lbin@raspberrypi:~ $ uname -a
Linux raspberrypi 4.19.128-rt55-v7l+ #1 SMP Thu May 27 15:18:00 UTC 2021 armv7l GNU/Linux
```

-> tout est OK \o\ /o/ <o/ \o> \o/

Pour finir je renomme l'image de Xenomai :

```
lbin@raspberrypi:~ $ sudo mv /boot/kernel7l.img /boot/kernel7l_xenomai.img
```

28 mai 2021 - labo D : installation du kernel 4.19.128

1. Préparation

Comme pour le projet précédent, je repars de la branche rpi-4.19.y de raspberry mergée avec le version 128 du kernel pour créer la branche rpi-4.19.y-kernel dans la projet labd-kernel/linux :

Je repars donc du repository du projet précédent et pour éviter de re télécharger tous les fichiers, je crée un nouvel espace de travail lié au repository existant :

```
~/2021_TR/labd-xenomai/rpi-kernel/linux$ git worktree add -b rpi-4.19.y-kernel
../../../labd-kernel/linux rpi-4.19.y
```

Sortie :

```
Preparing ../../../labd-kernel/linux (identifier linux1)
Checking out files: 100% (62377/62377), done.
HEAD is now at 1e5eb735d807 Merge tag 'v4.19.128' into rpi-4.19.y
```

2. Configuration

Création du répertoire contenant les fichiers compilés :

```
~/2021_TR/labd-kernel$ mkdir build-kernel
```

Configuration des variables :

```
export ARCH=arm
export INSTALL_MOD_PATH=~/2021_TR/labd-kernel/build-kernel
export INSTALL_DTBS_PATH=~/2021_TR/labd-kernel/build-kernel
export CROSS_COMPILE=arm-linux-gnueabihf-
export KERNEL=kernel7l
```

Application de la configuration :

```
~/2021_TR/labd-kernel/linux$ make bcm2711_defconfig
```

3. Compilation du kernel

```
~/2021_TR/labd-kernel/linux$ make -j4 zImage modules dtbs
~/2021_TR/labd-kernel/linux$ make -j4 modules_install dtbs_install
```

Sortie :

```
[...]
DEPMOD  4.19.128-v7l+
```

Création de l'image du kernel :

```
~/2021_TR/labd-kernel/linux$ mkdir $INSTALL_MOD_PATH/boot
~/2021_TR/labd-kernel/linux$ ./scripts/mkknlimg ./arch/arm/boot/zImage
$INSTALL_MOD_PATH/boot/$KERNEL.img
```

Sortie :

```
Version: Linux version 4.19.128-v7l+ (lbin@tr-ubuntu18-server) (gcc version 7.5.0
(Ubuntu/Linaro 7.5.0-3ubuntu1~18.04)) #1 SMP Fri May 28 08:20:37 UTC 2021
```

```
DT: y
DDT: y
270x: y
283x: y
```

Compression et copie de l'image sur le raspberry :

```
~/2021_TR/labd-kernel/build-kernel$ tar czf ../build-kernel.tgz *
~/2021_TR/labd-kernel$ scp build-kernel.tgz lbin@192.168.188.47:
```

4. Installation de l'image sur le raspberry

Extraction de l'image dans le répertoire rt-kernel :

```
lbin@raspberrypi:~ $ mkdir kernel && cd kernel
lbin@raspberrypi:~/kernel $ tar xf ../build-kernel.tgz
```

Comparaison des fichiers :

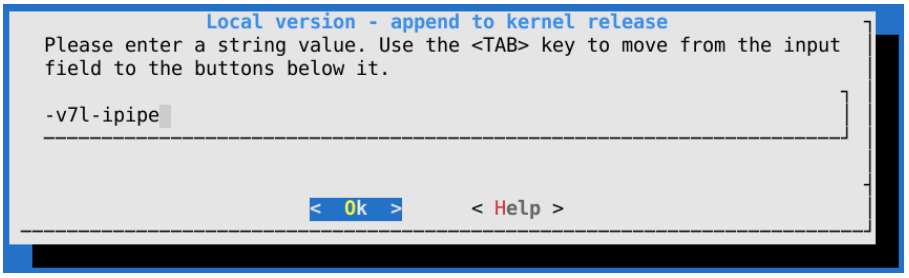
`~$ diff kernel/bcm2711-rpi-4-b.dtb xenomai/bcm2711-rpi-4-b.dtb` : pas de différence

`~/kernel/overlays $ for file in *; do diff "$file" "../../xenomai/overlays/$file"; done` : pas de différence

`~/kernel/lib/modules/4.19.128-v7l+ $ for file in **/**/*; do diff "$file" "../../../../xenomai/lib/modules/4.19.128-v7l+/$file"; done` : les fichiers sont différents et les répertoires ont le même nom -> problème

5. Recompilation de Xenomai

Je recompile le kernel avec Xenomai en changeant le nom (menuconfig : General setup -> Local version)




```
~/.../rpi-kernel/linux$ make -j4 zImage modules dtbs
~/.../rpi-kernel/linux$ make -j4 modules_install dtbs_install
```

sortie :

```
[...]
DEPMOD  4.19.128-v7l-ipipe+
```

Puis je recrée l'image et je la copie sur le raspberry de la même manière que la première fois et je renomme l'image :

```
lbin@raspberrypi:~/xenomai $ mv boot/kernel7l.img boot/kernel7l_xenomai.img
```

Copie de l'image et du répertoire /lib :

```
lbin@raspberrypi:~ $ sudo cp xenomai/boot/kernel7l_xenomai.img /boot/
lbin@raspberrypi:~ $ sudo cp -rd xenomai/lib/* /lib/
```

Edition du fichier /boot/config.txt :

```
kernel=kernel7l_xenomai.img
```

Au redémarrage :
lbin@raspberrypi:~ $ uname -a

```
Linux raspberrypi 4.19.128-v7l-ipipe+ #2 SMP Fri May 28 12:43:50 UTC 2021 armv7l GNU/Linux

lbin@raspberrypi:~ $ sudo /usr/xenomai/bin/latency
[sudo] password for lbin:
== Sampling period: 1000 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:01  (periodic user-mode task, 1000 us period, priority 99)
RTH|----lat min|----lat avg|----lat max|-overrun|---msw|---lat best|--lat worst
RTD|      0.666|      3.128|     23.777|       0|     0|      0.666|     23.777
RTD|      0.832|      2.059|     15.035|       0|     0|      0.666|     23.777
RTD|      1.128|      2.107|     12.869|       0|     0|      0.666|     23.777
```

6. Installation de l'image sur le raspberry sans tout casser

Copie de l'image et du répertoire /lib :

```
lbin@raspberrypi:~ $ sudo cp kernel/boot/kernel7l.img /boot/
lbin@raspberrypi:~ $ sudo cp -rd xenomai/lib/* /lib/
```

Edition du fichier /boot/config.txt :

```
kernel=${zImage name} #default: kernel 4.19.128
#kernel=kernel7l_rt.img
#kernel=kernel7l_xenomai.img
```

7. Redémarrage

```
lbin@raspberrypi:~ $ uname -a
Linux raspberrypi 4.19.128-v7l+ #1 SMP Fri May 28 08:20:37 UTC 2021 armv7l GNU/Linux
```

4 juin 2021 - labo E : periodic task

https://www.ashwinnarayan.com/post/xenomai-realtime-programming-part-2/
compilation directement sur le raspberry en me connectant par ssh via vs code

A la première compilation, j'ai un message d'erreur qui me dit que la commande xeno-config n'est pas trouvée. Je modifie donc le Makefile pour utiliser les variables qui y sont définies :

```
XENO_CONFIG=/usr/xenomai/bin/xeno-config
SKIN=alchemy
MAIN_SRC=cyclic_test
TARGET=cyclic_test

LM=-lm

CFLAGS := $(shell $(XENO_CONFIG) --skin=$(SKIN) --cflags)
LDFLAGS := $(LM) $(shell $(XENO_CONFIG) -- skin=$(SKIN) --ldflags)
CC := $(shell $(XENO_CONFIG) --cc)

$(TARGET): $(MAIN_SRC).c
    $(CC) -o $@ $< $(CFLAGS) $(LDFLAGS)
```

Le make fonctionne mais en lançant l'exécutable, j'ai un message d'erreur qui me dit que la librairie libalchemy n'est pas trouvée. Plutôt que de l'ajouter au path des librairies en settant la variable LD_LIBRARY_PATH, je crée le fichier xenomai.conf dans lequel j'ajoute le path de la librairie :

```
sudo mv /etc/ld.so.conf.d/libc.conf /etc/ld.so.conf.d/xenomai.conf
sudo vi /etc/ld.so.conf.d/xenomai.conf
       /usr/xenomai/lib
sudo ldconfig
```

Ensuite je peux compiler et lancer le programme :

```
lbin@raspberrypi:~/periodic_task $ make

gcc -o cyclic_test cyclic_test.c -I/usr/xenomai/include/cobalt -I/usr/xenomai/include
-D_GNU_SOURCE -D_REENTRANT -fasynchronous-unwind-tables -D__COBALT__
-I/usr/xenomai/include/alchemy -lm -Wl,--no-as-needed -Wl,@/usr/xenomai/lib/modechk.wrappers
-lalchemy -lcopperplate /usr/xenomai/lib/xenomai/bootstrap.o -Wl,--wrap=main
-Wl,--dynamic-list=/usr/xenomai/lib/dynlist.ld -L/usr/xenomai/lib -lcobalt -lmodechk -lpthread
-lrt

lbin@raspberrypi:~/periodic_task $ sudo ./cyclic_test

Starting cyclic task...
Starting task cyclic_task with period of 10 ms ....
Loop count: 0, Loop time: 0.00196 ms
Loop count: 1, Loop time: 9.95967 ms
Loop count: 2, Loop time: 19.95237 ms
Loop count: 3, Loop time: 29.95446 ms
Loop count: 4, Loop time: 39.94913 ms
Loop count: 5, Loop time: 49.95172 ms
Loop count: 6, Loop time: 59.95133 ms
Loop count: 7, Loop time: 69.95206 ms
Loop count: 8, Loop time: 79.94806 ms
Loop count: 9, Loop time: 89.94843 ms
[...]
```

7-8 juin 2021 - labo E : kernel module

Compilation du projet : tentative 1

https://github.com/harcokuppens/xenomai3_rpi_gpio/tree/master/examples/xenomai3/kernel_modules/test-irq-in-kernel-mode-using-gpiolib

Copie des sources sur le raspberry (selon la carte SD, ça peut prendre du temps) :

```
lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel$ tar czf xenomai-src.tar.gz linux

lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel$ scp xenomai-src.tar.gz
lbin@192.168.188.47:

lbin@raspberrypi:~/xenomai-src $ tar xf ../xenomai-src.tar.gz
```

Compilation :

```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm KERNEL=kernel7l -C /home/lbin/xenomai-src/linux M=/home/lbin/kernel_module
modules

make[1]: Entering directory '/home/lbin/xenomai-src/linux'
  CC [M]  /home/lbin/kernel_module/button_toggles_led.o
/home/lbin/kernel_module/button_toggles_led.c:7:10: fatal error: cobalt/kernel/ancillaries.h:
No such file or directory
 #include <cobalt/kernel/ancillaries.h>
          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~
compilation terminated.
make[2]: *** [scripts/Makefile.build:310: /home/lbin/kernel_module/button_toggles_led.o] Error
1
make[1]: *** [Makefile:1527: _module_/home/lbin/kernel_module] Error 2
make[1]: Leaving directory '/home/lbin/xenomai-src/linux'
make: *** [Makefile:25: modules] Error 2
```

Le compilateur ne trouve pas les fichiers d'include alors qu'ils sont bien présents dans les sources du kernel :

```
lbin@raspberrypi:~ $ find xenomai-src/linux/ -wholename *cobalt/kernel/ancillaries.h
xenomai-src/linux/include/xenomai/cobalt/kernel/ancillaries.h
```

En fait il s'agit de liens symboliques qui pointent vers un chemin dans la VM où le kernel a été compilé :

```
lbin@raspberrypi:~/xenomai-src/linux/include/xenomai $ ls -l
total 12
drwxr-xr-x 4 lbin lbin 4096 May 24 18:01 cobalt
drwxr-xr-x 4 lbin lbin 4096 May 24 18:01 rtdm
lrwxrwxrwx 1 lbin lbin   80 May 24 18:01 version.h ->
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/xenomai-3.1/include/xenomai/version.h
```

Compilation du projet : tentative 2

J'ai essayé de rajouter les chemins des includes dans le makefile mais en plus d'être une solution pas très élégante, c'était assez pénible de devoir rajouter le path de chaque fichier (ou presque) individuellement et j'ai donc abandonné :

```
EXTRA_CFLAGS := -I/home/lbin/xenomai-src/xenomai-3.1/include
-I/home/lbin/xenomai-src/xenomai-3.1/kernel/cobalt/arch/arm/include
-I/home/lbin/xenomai-src/xenomai-3.1/kernel/cobalt/include
-I/home/lbin/xenomai-src/xenomai-3.1/include/cobalt/kernel
```

etc …

Deux solutions sont possible : cross-compiler le code de kernel_module sur la VM sur laquelle le kernel a été compilé ou recompiler le kernel sur le raspberry. Je choisis la deuxième option.

Compilation du projet : tentative 3

Je réapplique le patch xenomai :

```
./xenomai-3.1/scripts/prepare-kernel.sh --linux=linux/ --ipipe=ipipe-core-4.19.128-arm-9.patch
--verbose
```

Les liens symboliques sont recréés correctement :

```
lbin@raspberrypi:~/xenomai-src $ ls -l linux/include/xenomai/
total 12
drwxr-xr-x 4 lbin lbin 4096 May 24 18:01 cobalt
drwxr-xr-x 4 lbin lbin 4096 Jun  7 19:41 rtdm
lrwxrwxrwx 1 lbin lbin   60 Jun  7 19:41 version.h ->
/home/lbin/xenomai-src/xenomai-3.1/include/xenomai/version.h
```

Je peux enlever l'infâme ligne d'EXTRA_CFLAGS mais la compilation ne marche toujours pas…

```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm KERNEL=kernel71 -C /home/lbin/xenomai-src/linux M=/home/lbin/kernel_module
CFLAGS="-I/home/lbin/xenomai-src/xenomai-3.1/include" modules
make[1]: Entering directory '/home/lbin/xenomai-src/linux'
  Building modules, stage 2.
  MODPOST 1 modules
/bin/sh: 1: scripts/mod/modpost: Exec format error
make[2]: *** [scripts/Makefile.modpost:92: __modpost] Error 2
make[1]: *** [Makefile:1531: modules] Error 2
make[1]: Leaving directory '/home/lbin/xenomai-src/linux'
make: *** [Makefile:25: modules] Error 2
```

Compilation du projet : tentative 4

Après avoir recompilé le kernel (c'est évidemment beaucoup plus lent sur le raspberry que sur la VM), je relance la compilation :

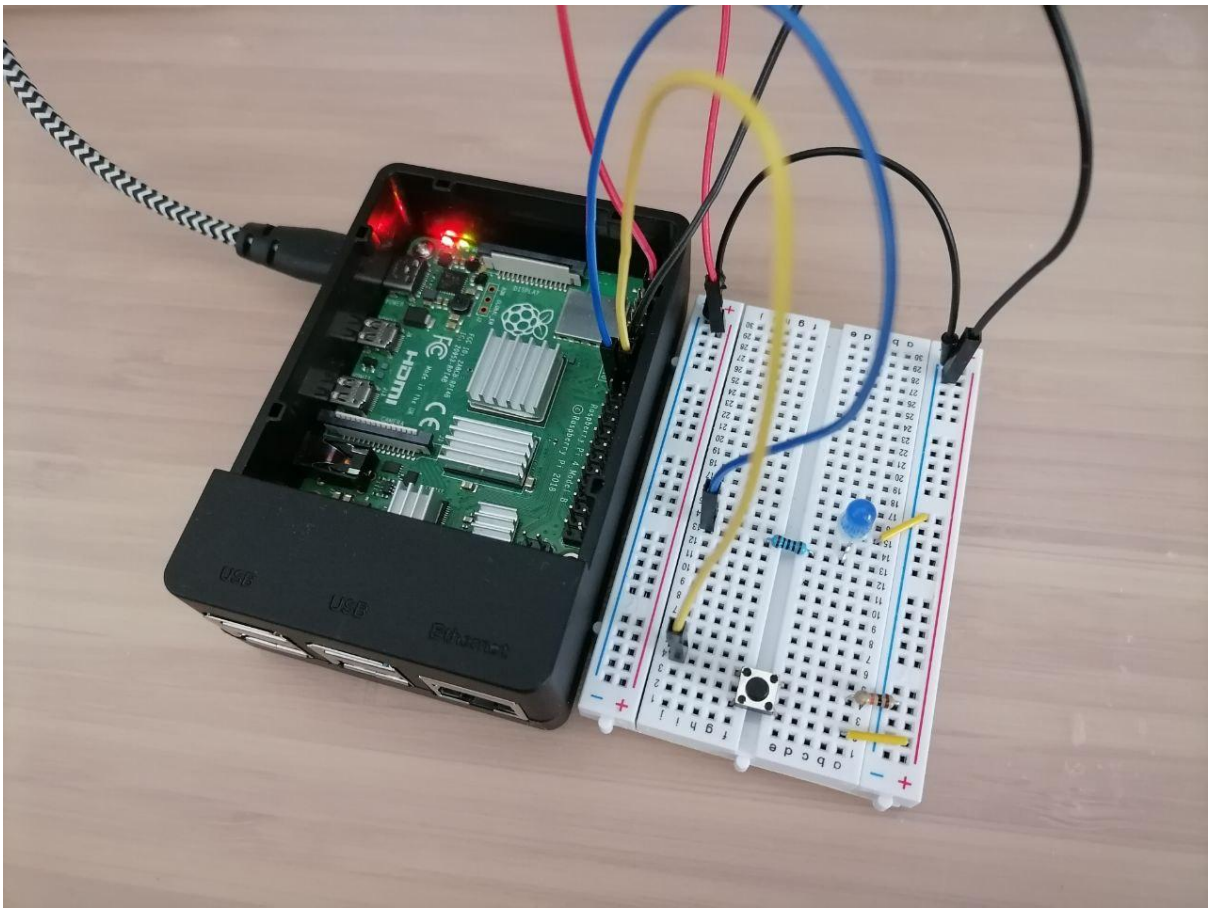
```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm  -C /home/lbin/xenomai-src/linux M=/home/lbin/kernel_module modules
make[1]: Entering directory '/home/lbin/xenomai-src/linux'
  Building modules, stage 2.
  MODPOST 1 modules
  CC    /home/lbin/kernel_module/button_toggles_led.mod.o
  LD [M]  /home/lbin/kernel_module/button_toggles_led.ko
make[1]: Leaving directory '/home/lbin/xenomai-src/linux'
```

Cette fois ça fonctionne \o/

```
lbin@raspberrypi:~ $ sudo insmod ./button_toggles_led.ko
lbin@raspberrypi:~/kernel_module $ lsmod
Module                  Size  Used by
[...]
button_toggles_led     16384  0
[...]
lbin@raspberrypi:~/kernel_module $ sudo rmmod ./button_toggles_led.ko
```

Interlude électronique : montage sur breadboard

Je monte un petit circuit sur breadboard avec un push button connecté à la GPIO23 et une LED connectée à la GPIO22 (la LED et le bouton sont donc indépendants). En appuyant puis en relâchant le push button, une interruption devrait être détectée par le système sur la GPIO23 et changer l'état de la GPIO22 pour allumer ou éteindre la LED.



- câble jaune : GPIO23 connectée au push button (résistance de 10k)
- câble bleu : GPIO22 connectée à la LED (résistance de 100 ohm)
- câble rouge : 5V
- câble noir : ground

Pour être sûre que tout fonctionne, j'ai testé toutes les connections et j'ai également testé l'allumage de la LED :

```
lbin@raspberrypi:~/kernel_module $ sudo -s
root@raspberrypi:/home/lbin/kernel_module# cd /sys/class/gpio/
root@raspberrypi:/sys/class/gpio# echo 22 >/sys/class/gpio/export
root@raspberrypi:/sys/class/gpio# ls
export    gpio22    gpiochip0  gpiochip504    unexport
root@raspberrypi:/sys/class/gpio# echo out >/sys/class/gpio/gpio22/direction
root@raspberrypi:/sys/class/gpio# echo 1 >/sys/class/gpio/gpio22/value
root@raspberrypi:/sys/class/gpio# echo 0 >/sys/class/gpio/gpio22/value
root@raspberrypi:/sys/class/gpio# echo 22 >/sys/class/gpio/unexport
root@raspberrypi:/sys/class/gpio# ls
export    gpiochip0  gpiochip504    unexport
root@raspberrypi:/sys/class/gpio# exit
exit
```

Test du module

Lorsque le module est lancé, la LED s'allume. Lorsque j'appuie sur le bouton, la LED change bien d'état sur le relâchement du bouton (le circuit de test étant très sommaire, il y a quelques ratés mais ça fonctionne globalement bien).

Par contre, je n'ai pas les logs :

```
lbin@raspberrypi:~/kernel_module $ sudo tail -11 /var/log/messages
Jun  8 07:29:57 raspberrypi lightdm[3565]: Error getting user list from
org.freedesktop.Accounts: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name
org.freedesktop.Accounts was not provided by any .service files
Jun  8 07:29:57 raspberrypi lightdm[3565]: Could not enumerate user data directory
/var/lib/lightdm/data: Error opening directory '/var/lib/lightdm/data': No such file or
directory
Jun  8 07:29:58 raspberrypi kernel: [  651.584055] broken atomic modeset userspace detected,
disabling atomic
Jun  8 07:29:59 raspberrypi kernel: [  652.752158] broken atomic modeset userspace detected,
disabling atomic
Jun  8 07:30:00 raspberrypi lightdm[3574]: Error getting user list from
org.freedesktop.Accounts: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name
org.freedesktop.Accounts was not provided by any .service files
Jun  8 07:30:00 raspberrypi lightdm[3574]: Could not enumerate user data directory
/var/lib/lightdm/data: Error opening directory '/var/lib/lightdm/data': No such file or
directory
Jun  8 07:30:01 raspberrypi kernel: [  654.367181] broken atomic modeset userspace detected,
disabling atomic
Jun  8 07:30:02 raspberrypi kernel: [  655.532409] broken atomic modeset userspace detected,
disabling atomic
Jun  8 07:30:02 raspberrypi lightdm[3583]: Error getting user list from
org.freedesktop.Accounts: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name
org.freedesktop.Accounts was not provided by any .service files
Jun  8 07:30:03 raspberrypi lightdm[3583]: Could not enumerate user data directory
/var/lib/lightdm/data: Error opening directory '/var/lib/lightdm/data': No such file or
directory
Jun  8 07:30:04 raspberrypi kernel: [  657.096278] broken atomic modeset userspace detected,
disabling atomic
```

Lorsque je lance dmesg -w, rien n'apparaît quand j'appuie sur la bouton :

```
lbin@raspberrypi:~/kernel_module $ dmesg -w
```

Fix des logs

https://stackoverflow.com/questions/8090944/printk-not-working-for-kernel-debgugging
https://lwn.net/Articles/487437/

Je remplace les printk par des pr_notice (et pr_emerg pour les erreurs) puis je recompile :

```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm  -C /home/lbin/xenomai-src/linux M=/home/lbin/kernel_module modules
make[1]: Entering directory '/home/lbin/xenomai-src/linux'
  CC [M]  /home/lbin/kernel_module/button_toggles_led.o
  Building modules, stage 2.
  MODPOST 1 modules
  LD [M]  /home/lbin/kernel_module/button_toggles_led.ko
make[1]: Leaving directory '/home/lbin/xenomai-src/linux'
```

En parallèle avec un terminal où j'ai lancé :

```
lbin@raspberrypi:~ $ dmesg -w
```

Je relance le module :

```
lbin@raspberrypi:~/kernel_module $ sudo insmod ./button_toggles_led.ko
```

Logs au lancement :

```
[10211.566531] button_toggle: irq number [54]
[10211.566550] button_toggle: cpu [3]
[10211.566566] button_toggle: GPIO23 request
[10211.566605] button_toggle: GPIO23 direction in
[10211.566627] button_toggle: GPIO22 request
[10211.566658] button_toggle: GPIO22 direction out
[10211.566679] button_toggle: GPIO23 ON
[10211.566697] button_toggle: irq trigger falling
[10211.566718] button_toggle: irq request
[10211.566755] button_toggle: globalCounter [0]
[10211.566770] button_toggle: handlerCPU [-1]
[10211.566784] button_toggle: init done
```

Quand j'appuie sur le bouton :

```
[ 9711.147637] button_toggle: interrupt
```

Quand j'arrête le module :

```
lbin@raspberrypi:~/kernel_module $ sudo rmmod button_toggles_led

[10251.396951] button_toggle: globalCounter [57]
[10251.396970] button_toggle: handlerCPU [0]
[10251.396985] button_toggle: exit done
```

Code source : labe_kernel_module.zip

9 juin - labo E : kernel module avec xenomai installé par dpkg

Je suis repartie de la sauvegarde que j'avais faite juste après avoir installé les trois kernels et compilé les outils Xenomai :

```
xzcat -v rt_backup.img.xz | sudo dd if=/dev/stdin of=/dev/mmcblk0 bs=1M
```

Installation du kernel Xenomai avec dpkg

Compilation des .deb depuis le répertoire dans lequel le kernel avec patch xenomai a déjà été compilé pour ne pas devoir recompiler complètement le kernel :

```
lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel/linux$ make oldconfig
lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel/linux$ make prepare
lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel/linux$ make -j4 deb-pkg
```

sur l'utilisation de oldconfig :
https://stackoverflow.com/questions/4178526/what-does-make-oldconfig-do-exactly-in-the-linux-kernel-makefile

Compression et envoi des .deb générés sur le raspberry :

```
lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel$ tar czf xenomai-img.tgz
linux-image-4.19.128-v7l-ipipe+_4.19.128-v7l-ipipe+-2_armhf.deb

lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel$ tar czf xenomai-hdr.tgz
linux-headers-4.19.128-v7l-ipipe+_4.19.128-v7l-ipipe+-2_armhf.deb

lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel$ scp xenomai-img.tgz
lbin@192.168.188.47:

lbin@tr-ubuntu18-server:~/2021_TR/labd-xenomai/rpi-kernel$ scp xenomai-hdr.tgz
lbin@192.168.188.47:
```

Installation sur le raspberry :

```
lbin@raspberrypi:~/xenomai $ tar xf ../xenomai-img.tgz

lbin@raspberrypi:~/xenomai $ tar xf ../xenomai-hdr.tgz

lbin@raspberrypi:~/xenomai $ sudo dpkg -i
linux-image-4.19.128-v7l-ipipe+_4.19.128-v7l-ipipe+-2_armhf.deb

lbin@raspberrypi:~/xenomai $ sudo dpkg -i
linux-headers-4.19.128-v7l-ipipe+_4.19.128-v7l-ipipe+-2_armhf.deb
```

Edition de /boot/config.txt :

```
#kernel=${zImage name} #default: kernel 4.19.128
#kernel=kernel7l_rt.img
kernel=vmlinuz-4.19.128-v7l-ipipe+
#kernel=kernel7l_xenomai.img
```

Reboot :

```
lbin@raspberrypi:~ $ uname -a
Linux raspberrypi 4.19.128-v7l-ipipe+ #2 SMP Fri May 28 12:43:50 UTC 2021 armv7l GNU/Linux
```

Compilation de kernel_module : tentative 1

```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm  -C /lib/modules/4.19.128-v7l-ipipe+/build M=/home/lbin/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
  CC [M]  /home/lbin/kernel_module/button_toggles_led.o
/home/lbin/kernel_module/button_toggles_led.c:7:10: fatal error: rtdm/driver.h: No such file
or directory
 #include <rtdm/driver.h> // replaces #include <rtdm/rtdm_driver.h> in xenomai 2
          ^~~~~~~~~~~~~~~
compilation terminated.
make[2]: *** [scripts/Makefile.build:310: /home/lbin/kernel_module/button_toggles_led.o] Error
1
make[1]: *** [Makefile:1527: _module_/home/lbin/kernel_module] Error 2
make[1]: Leaving directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
make: *** [Makefile:26: modules] Error 2
```

Le fichier se trouve bien dans les headers mais ne pointe pas au bon endroit :

```
lbin@raspberrypi:~/xenomai $ dpkg -c
linux-headers-4.19.128-v7l-ipipe+_4.19.128-v7l-ipipe+-2_armhf.deb | grep "rtdm/driver.h"
lrwxrwxrwx root/root          0 2021-05-24 18:01
./usr/src/linux-headers-4.19.128-v7l-ipipe+/include/xenomai/rtdm/driver.h ->
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/xenomai-3.1/include/cobalt/kernel/rtdm/driver.h
```

Pour réparer le lien symbolique, je déplace les sources de xenomai au même endroit qu'indiqué par le lien :

```
lbin@raspberrypi:~ $ mkdir -p  2021_TR/labd-xenomai/rpi-kernel/xenomai-3.1
lbin@raspberrypi:~ $ cp -ar xenomai-3.1/* 2021_TR/labd-xenomai/rpi-kernel/xenomai-3.1/

lbin@raspberrypi:~ $ ls
/usr/src/linux-headers-4.19.128-v7l-ipipe+/include/xenomai/rtdm/driver.h

/usr/src/linux-headers-4.19.128-v7l-ipipe+/include/xenomai/rtdm/driver.h
```

Compilation de kernel_module : tentative 2

```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm  -C /lib/modules/4.19.128-v7l-ipipe+/build M=/home/lbin/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
  CC [M]  /home/lbin/kernel_module/button_toggles_led.o
In file included from include/xenomai/cobalt/kernel/thread.h:27,
                 from include/xenomai/cobalt/kernel/sched.h:24,
                 from include/xenomai/rtdm/driver.h:37,
                 from /home/lbin/kernel_module/button_toggles_led.c:7:
include/xenomai/cobalt/kernel/timer.h:28:10: fatal error: asm/xenomai/wrappers.h: No such file
or directory
 #include <asm/xenomai/wrappers.h>
          ^~~~~~~~~~~~~~~~~~~~~~~~
compilation terminated.
make[2]: *** [scripts/Makefile.build:310: /home/lbin/kernel_module/button_toggles_led.o] Error
1
make[1]: *** [Makefile:1527: _module_/home/lbin/kernel_module] Error 2
make[1]: Leaving directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
make: *** [Makefile:26: modules] Error 2
```

Ajout de flags pour inclure le chemin où trouver ce fichier :

```
EXTRA_CFLAGS := -I/home/lbin/xenomai-3.1/kernel/cobalt/arch/arm/include
(EXTRA_CFLAGS := $(shell $(XENO_CONFIG) --rtdm --cflags)
-I/home/lbin/xenomai-3.1/kernel/cobalt/arch/arm/include)
```

https://xenomai.org/pipermail/xenomai/2015-March/033555.html
https://xenomai.org/pipermail/xenomai-git/2015-August/005238.html

Compilation de kernel_module : tentative 3

```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm  -C /lib/modules/4.19.128-v7l-ipipe+/build M=/home/lbin/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
  CC [M]  /home/lbin/kernel_module/button_toggles_led.o
/bin/sh: 1: scripts/basic/fixdep: Exec format error
make[2]: *** [scripts/Makefile.build:310: /home/lbin/kernel_module/button_toggles_led.o] Error
2
make[1]: *** [Makefile:1527: _module_/home/lbin/kernel_module] Error 2
make[1]: Leaving directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
make: *** [Makefile:26: modules] Error 2
```

Problème dû à la cross-compilation : fixdep ne s'exécute pas. Il y a un bug lors de la cross-compilation et on trouve ici comment le résoudre.

Application d'un patch :

```
lbin@raspberrypi:/usr/src/linux-headers-4.19.128-v7l-ipipe+ $ wget
https://raw.githubusercontent.com/armbian/build/master/patch/misc/headers-debian-byteshift.pat
ch -O - | sudo patch -p1
[...]
patching file tools/include/tools/be_byteshift.h
patching file tools/include/tools/le_byteshift.h
```

Recompilation de tous les scripts :

```
lbin@raspberrypi:/usr/src/linux-headers-4.19.128-v7l-ipipe+ $ sudo make scripts
  HOSTCC  scripts/basic/fixdep
  [...]
```

-> fixdep a bien été recompilé

Compilation de kernel_module : tentative 4

```
lbin@raspberrypi:~/kernel_module $ make
make ARCH=arm  -C /lib/modules/4.19.128-v7l-ipipe+/build M=/home/lbin/kernel_module modules
make[1]: Entering directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
  Building modules, stage 2.
  MODPOST 1 modules
  CC    /home/lbin/kernel_module/button_toggles_led.mod.o
  LD [M]  /home/lbin/kernel_module/button_toggles_led.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.19.128-v7l-ipipe+'
```

Cette fois tout fonctionne \o/

Test de kernel_module

Je refais les mêmes tests que précédemment en lançant le module dans un terminal en parallèle avec un terminal où j'affiche les logs du kernel :

```
lbin@raspberrypi:~ $ dmesg -w
lbin@raspberrypi:~/kernel_module $ sudo insmod ./button_toggles_led.ko
```

Logs au lancement :

```
[ 7641.127551] button_toggles_led: loading out-of-tree module taints kernel.
[ 7641.129580] button_toggle: irq number [54]
[ 7641.129587] button_toggle: cpu [1]
[ 7641.129595] button_toggle: GPIO23 request
[ 7641.129616] button_toggle: GPIO23 direction in
[ 7641.129627] button_toggle: GPIO22 request
[ 7641.129642] button_toggle: GPIO22 direction out
[ 7641.129651] button_toggle: GPIO23 ON
[ 7641.129658] button_toggle: irq trigger falling
[ 7641.129667] button_toggle: irq request
[ 7641.129686] button_toggle: globalCounter [0]
[ 7641.129694] button_toggle: handlerCPU [-1]
[ 7641.129702] button_toggle: init done
```

Quand j'appuie sur le bouton :

```
[ 7687.844816] button_toggle: interrupt
```

Quand j'arrête le module :

```
lbin@raspberrypi:~/kernel_module $ sudo rmmod button_toggles_led

[ 7724.169708] button_toggle: globalCounter [79]
[ 7724.169720] button_toggle: handlerCPU [0]
[ 7724.169729] button_toggle: exit done
```

Code source (changements dans le Makefile de la variable KERNELDIR et ajout des EXTRA_CFLAGS) : labe_kernel_module_v2.zip

10 juin - labo E : cross compilation de kernel module

Pour la cross compilation, j'ai juste changé la variable KERNELDIR du Makefile :

```
KERNELDIR ?=
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/rt-kernel/lib/modules/4.19.128-v7l-ipipe+/build
```

Et j'ai rajouté les outils de cross compilation et l'architecture de destination :

```
$(MAKE) ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C $(KERNELDIR) M=$(PWD) modules
```

Puis je compile et tout fonctionne :

```
lbin@tr-ubuntu18-server:~/2021_TR/labe_kernel_module$ make
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -C
/home/lbin/2021_TR/labd-xenomai/rpi-kernel/rt-kernel/lib/modules/4.19.128-v7l-ipipe+/build
M=/home/lbin/2021_TR/labe_kernel_module modules
make[1]: Entering directory '/home/lbin/2021_TR/labd-xenomai/rpi-kernel/linux'
  CC [M]  /home/lbin/2021_TR/labe_kernel_module/button_toggles_led.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC    /home/lbin/2021_TR/labe_kernel_module/button_toggles_led.mod.o
  LD [M]  /home/lbin/2021_TR/labe_kernel_module/button_toggles_led.ko
make[1]: Leaving directory '/home/lbin/2021_TR/labd-xenomai/rpi-kernel/linux'
```

J'envoie le projet sur le raspberry :

```
lbin@tr-ubuntu18-server:~/2021_TR$ tar cfz labe_kernel_module.tgz labe_kernel_module/*

lbin@tr-ubuntu18-server:~/2021_TR$ scp labe_kernel_module.tgz lbin@192.168.188.47:
```

Je décompresse l'archive :

```
lbin@raspberrypi:~ $ tar xf labe_kernel_module.tgz
lbin@raspberrypi:~ $ cd labe_kernel_module/
lbin@raspberrypi:~/labe_kernel_module $ ls
button_toggles_led.c   button_toggles_led.mod.o  modules.order
button_toggles_led.ko  button_toggles_led.o   Module.symvers
button_toggles_led.mod.c  Makefile
```

Puis je teste le module sur le raspberry comme précédemment en lançant en parallèle :

```
lbin@raspberrypi:~ $ dmesg -w
lbin@raspberrypi:~/labe_kernel_module $ sudo insmod ./button_toggles_led.ko
```

Logs au lancement :

```
[ 2666.304590] button_toggles_led: loading out-of-tree module taints kernel.
[ 2666.305036] button_toggle: irq number [54]
[ 2666.305043] button_toggle: cpu [1]
[ 2666.305049] button_toggle: GPIO23 request
[ 2666.305066] button_toggle: GPIO23 direction in
[ 2666.305075] button_toggle: GPIO22 request
[ 2666.305088] button_toggle: GPIO22 direction out
[ 2666.305097] button_toggle: GPIO23 ON
[ 2666.305105] button_toggle: irq trigger falling
[ 2666.305113] button_toggle: irq request
[ 2666.305127] button_toggle: globalCounter [0]
[ 2666.305133] button_toggle: handlerCPU [-1]
[ 2666.305139] button_toggle: init done
```

Quand j'appuie sur le bouton :

```
[ 2691.786060] button_toggle: interrupt
```

Quand j'arrête le module :

```
lbin@raspberrypi:~/labe_kernel_module $ sudo rmmod button_toggles_led

[ 2723.635037] button_toggle: globalCounter [1]
[ 2723.635049] button_toggle: handlerCPU [0]
[ 2723.635058] button_toggle: exit done
```

Code source : labe_kernel_module_v3_cross_compilation.zip