

UE B0 - Systèmes multitâches

Compte rendu de TP Systèmes multitâches

THOMAS BOULANGER JULES FARNAULT

Table des matières

Table des matières Introduction		1 2	
			1
2	Premier exercice (Exercice Principal)		5
	2.1	Conception détaillée en utilisant une approche dirigée par les événements .	6
	2.2	Implémentation dirigée par les événements	9
	2.3	Pour aller plus loin en conception et implémentation dirigées par les évé-	
		nements	11
	2.4	Conception détaillée en utilisant une approche dirigée par le temps	15
C	Conclusion		

Introduction

Les objectifs de ce TP sont multiples :

- Concevoir des applications multitâche (non temps réel) avec une approche dirigée par les événements
- Manipuler les mécanismes POSIX d'un système d'exploitation Linux

et

- Manipuler les mécanismes C11
- Concevoir une application multitâche avec une approche dirigée par le temps

1 Préambule

L'objectif de cette partie est de prendre en main la création d'une tâche et d'un sémaphore en utilisant les APIs POSIX.

On ouvre preambule.c dans Vim. On identifie alors le nom du thread (thread_1), du sémaphore ("/preambule_sem") et du mutex (PTHREAD_MUTEX_INITIALIZER;) à partir de la ligne 10:

```
#define SEM_NAME "/preambule_sem"

pthread_t thread_1;
sem_t *semaphore;

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

La création du thread se fait à la ligne 44, son point d'entrée est la fonction *produce* :

```
pthread_create(&thread_1, NULL, produce, NULL);
```

L'attente de la fin du thread se fait à la ligne 47 :

```
pthread_join(thread_1, NULL);
```

Le thread par défaut du processus courant est la fonction main. Dans notre programme, nous avons deux process : le terminal et le main.

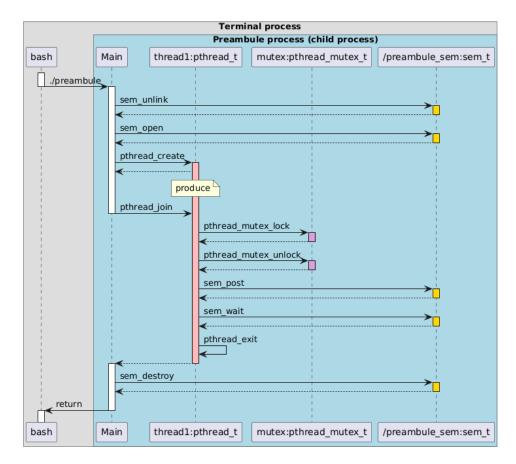


FIGURE 1.1 – Diagramme UML de l'exécution de preambule.c

Même si l'on considère qu'il n'y a pas d'exigences temporelles, la conception de ce programme n'est pas complète, car seul le cas nominal est représenté, il manque les cas d'erreurs (erreur de création de sémaphore, de mutex, etc).

2 Premier exercice (Exercice Principal)

Le but de cette partie est de développer un accumulateur (MultitaskingAccumulator) qui réalise l'acquisition de quatre entrées numériques asynchrones auprès d'un composant logiciel externe (SensorManager) et qui somme ces entrées. Les exigences sont définies dans l'énoncé.

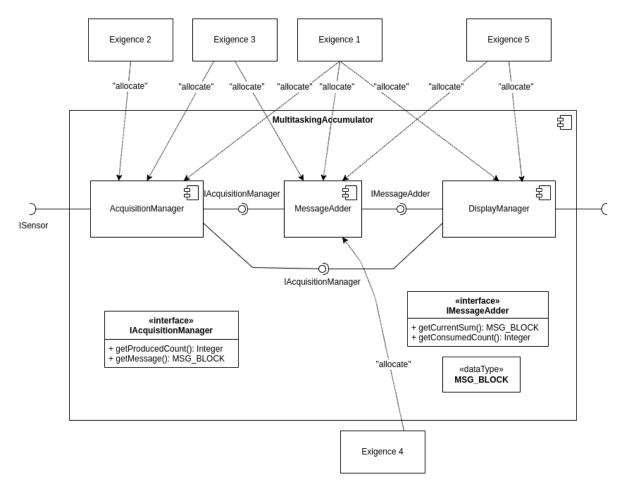


FIGURE 2.1 – Allocation des exigences du système

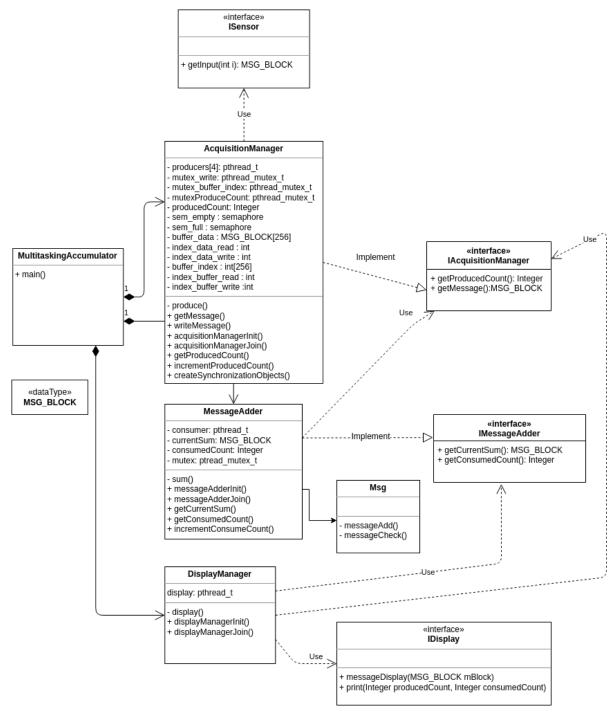
2.1 - Conception détaillée en utilisant une approche dirigée par les événements

Une approche dirigée par les événements est une approche asynchrone, on agit seulement quand un événement apparaît et ils n'apparaissent pas nécessairement de façon périodique.

La méthode messageCheck vérifie la checksum du message, ce contrôle répond à une stratégie de *Sanity Checking* appartenant à la catégorie des *Failure Detection* et permet de s'assurer qu'il n'y a pas de corruption du message.

Le programme est composé d'un unique process généré dans le main de MultitaskingAccumulator.c. Ce programme contient 7 threads :

- Le thread du main
- 4 threads pour l'Acquisition manager
- 1 thread pour le MessageAdder
- 1 thread pour le DisplayManager



 ${\tt FIGURE~2.2-Architecture~logicielle~d\acute{e}taill\acute{e}e~de~la~conception}$

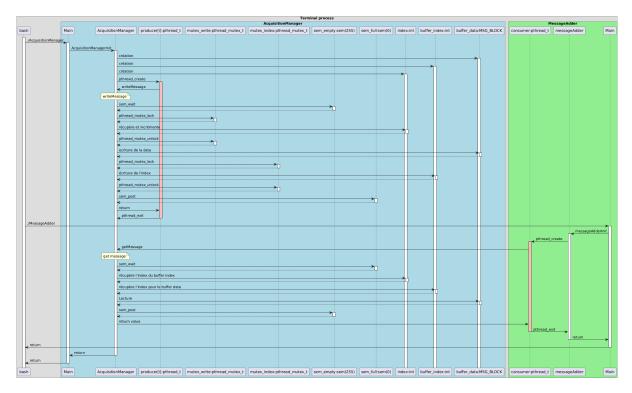


FIGURE 2.3 – Diagramme de séquence entre l'Acquisition manager et le MessageAdder

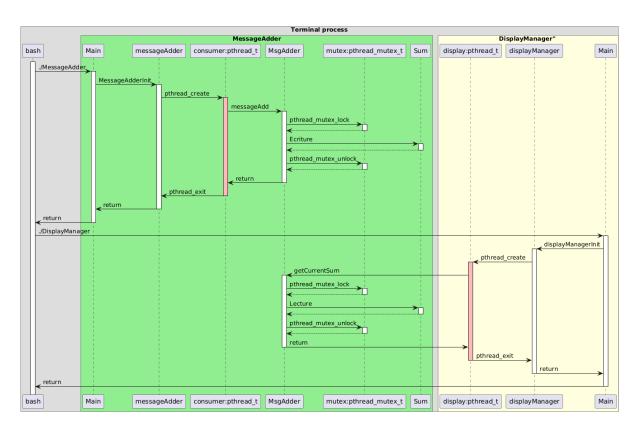


FIGURE 2.4 – Diagramme de séquence entre le MessageAdder et DisplayManager

2.2 - Implémentation dirigée par les événements

On implémente la conception précédente et on regarde sa bonne exécution :

```
cc -pthread -o multitaskingAccumulatorPosix acquisitionManagerPOSIX.o
    display.o displayManager.o messageAdder.o msg.o
    - multitaskingAccumulator.o sensorManager.o
    ./multitaskingAccumulatorPosix
    [multitaskingAccumulator] Software initialization in progress...
3
    [acquisitionManager] Synchronization initialization in progress...
    [acquisitionManager] Synchronization initialization done.
5
    [multitaskingAccumulator] Task initialization done.
6
    [multitaskingAccumulator]Scheduling in progress...
7
    [OK]
             ] Checksum validated
8
    ГОК
             ] Checksum validated
9
             ] Checksum validated
    [OK
10
    [OK]
             ] Checksum validated
11
             ] Checksum validated
    LOK
12
             l Checksum validated
    [OK
13
   Message
    [displayManager]Produced messages: 3, Consumed messages: 1, Messages
15
    → left: 2
    ГОК
             ] Checksum validated
16
    ГОК
             Checksum validated
17
    [acquisitionManager] 44019 termination
18
             ] Checksum validated
19
    [acquisitionManager] 44018 termination
    □OK
             ] Checksum validated
21
    [OK
             ] Checksum validated
22
    [acquisitionManager] 44017 termination
23
    ГОК
             Checksum validated
24
    [OK]
             ] Checksum validated
25
   Message
26
    [displayManager]Produced messages: 7, Consumed messages: 2, Messages
       left: 5
    LOK
             ] Checksum validated
28
    [OK
             ] Checksum validated
29
    [acquisitionManager] 44016 termination
30
             ] Checksum validated
    □OK
31
    [OK]
             ] Checksum validated
32
    [OK
             ] Checksum validated
   Message
34
    [displayManager] Produced messages: 8, Consumed messages: 4, Messages
35
    → left: 4
    ГОК
             ] Checksum validated
36
    ГОК
             l Checksum validated
37
    [OK
             ] Checksum validated
```

```
Message
39
    [displayManager]Produced messages: 8, Consumed messages: 5, Messages
40
    → left: 3
    \lceil \mathsf{OK} \rceil
             ] Checksum validated
41
             ] Checksum validated
    [OK
42
             ] Checksum validated
    ГОК
43
             ] Checksum validated
    ГОК
44
   Message
45
    [displayManager]Produced messages: 8, Consumed messages: 7, Messages
    → left: 1
    [OK
             ] Checksum validated
47
    [messageAdder] 44020 termination
48
             ] Checksum validated
    [OK
49
             ] Checksum validated
    [OK
50
   Message
51
    [displayManager]Produced messages: 8, Consumed messages: 8, Messages
    → left: 0
    ГОК
             ] Checksum validated
53
             ] Checksum validated
    [OK
54
```

La cohérence sur IDisplay est garanti par la création d'une nouvelle structure qui contient le compteur ConsumedCount et la somme. Ainsi, on obtient les deux variables en même temps en appelant la fonction getCurrentSum(), ce qui garanti la synchronisation entre les deux variables.

```
typedef struct MSG_BLOCK_TAG2

{
          MSG_BLOCK mBlock;
          unsigned int consumedCount;
} MSG_BLOCK_with_ConsumedCount;
```

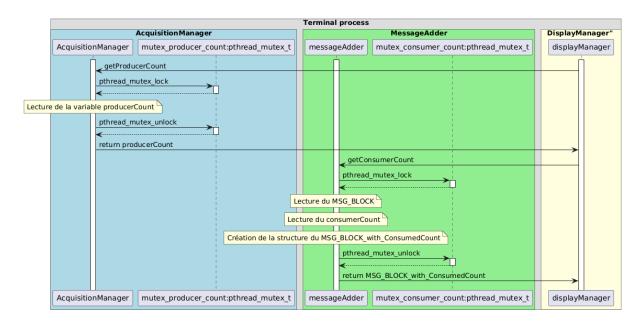


FIGURE 2.5 – Diagramme de séquence pour la cohérence d'information de l'exigence 5

Certaines variables sont considérées comme des variables C volatiles afin de désactiver les optimisations du compilateur sur les ressources partagé entre les threads, ce qui permet de garantir la cohérence spatiale des données.

2.3 - Pour aller plus loin en conception et implémentation dirigées par les événements

Les processus POSIX possèdent un partitionnement spatial qui permet d'ajouter une sécurité évitant la corruption des données. Cependant, entre les processus, il n'est pas possible d'utiliser des mutexs, il faut également utiliser de la mémoire partagée ou des pipes afin de synchroniser les données. La conception faite précédemment ne sera donc pas utilisable tel qu'elle, il faudra fortement la modifier pour qu'elle puisse répondre aux exigences avec des processus POSIX.

Afin de protéger le compteur permettant compter le nombre de messages produits de manière efficace entre les tâches et sans utiliser des apis POSIX, on peut utiliser le mot clé _atomic en C.

On peut alors retirer le mutex mutexProduceCount dédié à la manipulation de produceCount :

```
_Atomic volatile unsigned int produceCount = 0;

static void incrementProducedCount(void)

produceCount++;

produceCount++;

produceCount++;
```

```
unsigned int getProducedCount(void)

unsigned int p = 0;

p = produceCount;

return p;
}
```

On lance alors le programme avec cette modification, on a le résultat suivant :

```
./multitaskingAccumulatorAtomic
1
    [multitaskingAccumulator]Software initialization in progress...
2
    [acquisitionManager] Synchronization initialization in progress...
3
    [acquisitionManager]Synchronization initialization done.
    [multitaskingAccumulator] Task initialization done.
5
    [multitaskingAccumulator]Scheduling in progress...
6
             ] Checksum validated
    [OK
7
    [OK]
             ] Checksum validated
8
    LOK
             ] Checksum validated
9
    ГОК
             Checksum validated
10
    ГОК
             ] Checksum validated
11
    [OK
             ] Checksum validated
12
   Message
13
    [displayManager]Produced messages: 3, Consumed messages: 1, Messages
14
       left: 2
    LOK
             ] Checksum validated
15
    ГОК
             ] Checksum validated
16
    [acquisitionManager] 92686 termination
17
             ] Checksum validated
    [OK
18
    [acquisitionManager] 92687 termination
19
    ГОК
             l Checksum validated
20
             ] Checksum validated
21
    [acquisitionManager] 92685 termination
22
    LOK
             ] Checksum validated
23
             ] Checksum validated
    ГОК
   Message
25
    [displayManager]Produced messages: 7, Consumed messages: 2, Messages
26
       left: 5
    ГОК
             Checksum validated
27
    [OK
             ] Checksum validated
28
    [acquisitionManager] 92684 termination
29
             ] Checksum validated
    [OK
30
             ] Checksum validated
    [OK
31
    [OK
             ] Checksum validated
32
   Message
33
    [displayManager]Produced messages: 8, Consumed messages: 4, Messages
34
       left: 4
    [OK
             ] Checksum validated
```

```
\GammaOK
              ] Checksum validated
36
    \lceil \mathsf{OK} \rceil
              ] Checksum validated
37
    Message
    [displayManager]Produced messages: 8, Consumed messages: 5, Messages
39
     → left: 3
    ГОК
              ] Checksum validated
40
    ГОК
              Checksum validated
41
    \lceil \mathsf{OK} \rceil
              ] Checksum validated
42
    [OK]
              ] Checksum validated
43
    Message
44
    [displayManager]Produced messages: 8, Consumed messages: 7, Messages
45
        left: 1
    LOK
              ] Checksum validated
46
    [messageAdder] 92688 termination
47
              ] Checksum validated
48
    [OK
              ] Checksum validated
    Message
50
    [displayManager]Produced messages: 8, Consumed messages: 8, Messages
51
     → left: 0
    [OK
              ] Checksum validated
52
    LUK
              l Checksum validated
53
```

On va maintenant utiliser une approche Test and Set sur la variable ProduceCount. Pour cela, on utilise la fonction atomic_compare_exchange_weak qui permet de comparer une valeur attendue et de la changer si elles correspondent.

```
_Atomic volatile unsigned int produceCount = 0;
    _Atomic volatile int lock = 0;
2
3
   static void incrementProducedCount(void)
4
    {
5
            // produceCount++;
6
            int expected = 0;
7
            while (!atomic_compare_exchange(&lock, &expected, 1))
8
9
            }
10
            produceCount++;
11
            lock = 0;
12
   }
13
   unsigned int getProducedCount(void)
15
   {
16
            unsigned int p = 0;
17
            int expected = 0;
18
            while (!atomic_compare_exchange(&lock, &expected, 1))
19
            {
20
            }
```

```
p = produceCount;
lock = 0;
return p;
}
```

On exécute la commande make runtestandset et on observe le comportement du programme :

```
./multitaskingAccumulatorTestAndSet
1
    [multitaskingAccumulator]Software initialization in progress...
2
    [acquisitionManager] Synchronization initialization in progress...
3
    [acquisitionManager] Synchronization initialization done.
4
    [multitaskingAccumulator] Task initialization done.
    [multitaskingAccumulator]Scheduling in progress...
6
             ] Checksum validated
    「OK
7
             l Checksum validated
    ГОК
8
    [OK
             ] Checksum validated
9
    LOK
             ] Checksum validated
10
    ГОК
             Checksum validated
11
    ГОК
             ] Checksum validated
12
   Message
13
    [displayManager]Produced messages: 3, Consumed messages: 1, Messages
14
    → left: 2
    LOK
             Checksum validated
15
    [OK
             ] Checksum validated
16
    [acquisitionManager] 120571 termination
17
    ГОК
             ] Checksum validated
18
    [acquisitionManager] 120570 termination
19
    [OK
             ] Checksum validated
20
    ГОК
             l Checksum validated
21
    [acquisitionManager] 120569 termination
22
    \GammaOK
             ] Checksum validated
23
    ГОК
             ] Checksum validated
24
   Message
25
    [displayManager] Produced messages: 7, Consumed messages: 2, Messages
26
    → left: 5
    ГОК
             l Checksum validated
27
    ГОК
             ] Checksum validated
28
    [acquisitionManager] 120568 termination
29
    LOK
             ] Checksum validated
             ] Checksum validated
    [OK
31
    [OK
             ] Checksum validated
   Message
33
    [displayManager]Produced messages: 8, Consumed messages: 4, Messages
34
       left: 4
    LUK
             ] Checksum validated
35
    [OK
             ] Checksum validated
```

```
\GammaOK
              ] Checksum validated
37
   Message
38
    [displayManager]Produced messages: 8, Consumed messages: 5, Messages
        left: 3
    [OK
              ] Checksum validated
40
              l Checksum validated
    ΓOK
41
    ГОК
              Checksum validated
42
    \lceil \mathsf{OK} \rceil
              ] Checksum validated
43
   Message
    [displayManager]Produced messages: 8, Consumed messages: 7, Messages
45
        left: 1
    ГОК
              ] Checksum validated
46
    [messageAdder] 120572 termination
47
    ГОК
              ] Checksum validated
48
    ГОК
              l Checksum validated
49
   Message
    [displayManager]Produced messages: 8, Consumed messages: 8, Messages
51
        left: 0
    [OK
              ] Checksum validated
52
              ] Checksum validated
    [OK]
```

Les résultats présentés dans le sujet montre que le programme Atomic est le plus rapide par rapport à Posix et TestAndSet. Cela vient du fait que le programme atomique ne fait pas d'appel système (syscall) et donc de changement de contexte, ce qui réduit considérablement le temps d'exécution. La gestion de l'atomicité est gérée directement par le compilateur.

2.4 - Conception détaillée en utilisant une approche dirigée par le temps

Une approche dirigée par le temps est une approche synchrone. C'est-à-dire que l'on a une garantie temporelle sur la disponibilité des données

Conclusion

Ce TP nous a permis de découvrir de manière pratique le développement et la prise en compte de la sécurité des systèmes multitâche à travers diverse exemples (POSIX, Atomic, Test and set) en utilisant des diagrammes UML et des représentations de l'architecture logicielle.