



**POLYTECHNIQUE
MONTRÉAL**

UNIVERSITÉ
D'INGÉNIERIE

INF3610 – Systèmes embarqués

Automne 2024

TP No. 1 (Partie 1)

Groupe 02



Soumis à



Mardi 01 Octobre 2024

Table des matières	2
Question 1	3
Question 2	4
Question 3	9
Question 4	15

Questions pour le rapport

Question 1 En vous référant aux priorités assignées aux tâches et en analysant le code de TaskComputing, répondez aux questions suivantes :

a) Pourquoi durant la génération des paquets, le nombre d'éléments de fifo de TaskQueuing est toujours maintenu assez bas (proche de 0)?

Le nombre d'éléments dans le FIFO de TaskQueuing est maintenu bas car les tâches sont consommées rapidement par TaskComputing grâce à un traitement efficace et rapide. C'est crucial pour éviter la surcharge du FIFO et garantir que les paquets sont traités dès leur création, ce qui permet une gestion fluide des tâches en temps réel.

b) À la figure 10, l'affichage des fifos de TaskComputing HighQ et MediumQ (resp. 18 et 19 éléments) arrive après la fin de la rafale. Pourquoi ?

L'affichage après la rafale indique que le traitement des paquets a été effectué en plusieurs étapes, avec une priorisation des tâches HighQ et MediumQ. Cela est dû au mécanisme de gestion des priorités, où les tâches plus importantes (HighQ) sont traitées en premier, ce qui peut expliquer un décalage dans la visualisation après une rafale de création de paquets.

c) À la figure 11, pourquoi l'affichage du paquet reçu id 1 de type HighQ (venant de TaskOutputPort[0]) arrive avant la mise à jour de TaskComputing de HighQ (i.e. TaskComputing HighQ: nb de paquets après consommation du fifo: 17)?

À cause de l'ordre de traitement et d'affichage des informations, le système reçoit d'abord le paquet, puis le traite avant d'actualiser l'état du FIFO, ce qui crée un léger décalage temporel entre la réception du paquet et la mise à jour du FIFO.

d) Comment expliquez-vous que les fifos HighQ se vide toujours avant MediumQ et que MediumQ se vide toujours avant LowQ?

La hiérarchie des priorités assignées aux différentes files d'attente (HighQ > MediumQ > LowQ) explique pourquoi, les tâches les plus prioritaires (HighQ) sont toujours consommées en premier, suivies par les tâches MediumQ, puis les tâches LowQ.

e) Justifiez le choix de la priorité des tâches dans routeur.h. Selon vous, est-ce le meilleur choix, y a-t-il d'autres assignations possibles et comment le comportement serait alors modifié.

Le choix de prioriser HighQ, MediumQ et LowQ dans cet ordre est justifié dans un contexte temps réel où les tâches critiques doivent être traitées immédiatement. Cependant, d'autres assignations pourraient être envisagées, par exemple une gestion dynamique basée sur la charge des différentes files d'attente. Cela modifierait le comportement en permettant une répartition plus équitable des ressources, mais cela pourrait aussi augmenter le temps de réponse pour les tâches critiques.

Figure 1 : Manipulation 1

```
Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 128
Frequence du systeme: 8000
1 - Nb de packets total crees : 105103
2 - Nb de packets total traitees : 94251
3 - Nb de packets rejetes pour mauvaise source : 1317
4 - Nb de packets rejetes pour mauvaise source total: 8484
5 - Nb de packets rejetes pour mauvais CRC : 148
6 - Nb de packets rejetes pour mauvais CRC total : 903
7 - Nb de paquets rejetes fifo : 0
8 - Nb de paquets rejetes fifo total : 0
9 - Nb de paquets rejetes mauvaise priorites : 0
10 - Nb de paquets rejetes mauvaise priorites total : 0
11 - Nb de paquets maximum dans le fifo de Queueing : 251
12 - Nb de paquets maximum dans le fifo HIGHQ de TaskComputing: 92
13 - Nb de paquets maximum dans fifo MEDIUMQ de TaskComputing: 89
14 - Nb de paquets maximum dans fifo LOWQ de TaskComputing: 176
15 - Nb de paquets maximum dans port0 : 0
16 - Nb de paquets maximum dans port1 : 0
17 - Nb de paquets maximum dans port2 : 0

18- Message free : 5880
19- Message used : 120
20- Message used max : 298
21- Nombre de ticks depuis le début de l'execution 1687091
22- Pire temps video '0.0922401994'
23- Pire temps audio '0.1832783967'
24- Pire temps autre '0.4495934546'
----- Task stop suspend all tasks -----
----- Flags: 0 -----
```

 Send

Clear

Figure 2 : Manipulation 2

Connected to: Serial (COM6, 115200, 0, 8)

```
Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 128
Frequence du systeme: 8000
1 - Nb de packets total crees : 105103
2 - Nb de packets total traitees : 94251
3 - Nb de packets rejetees pour mauvaise source : 1316
4 - Nb de packets rejetees pour mauvaise source total: 8485
5 - Nb de packets rejetees pour mauvais CRC : 148
6 - Nb de packets rejetees pour mauvais CRC total : 903
7 - Nb de paquets rejetees fifo : 0
8 - Nb de paquets rejetees fifo total : 0
9 - Nb de paquets rejetees mauvaise priorites : 0
10 - Nb de paquets rejetees mauvaise priorites total : 0
11 - Nb de paquets maximum dans le fifo de Queueing : 251
12 - Nb de paquets maximum dans le fifo HIGHQ de TaskComputing: 92
13 - Nb de paquets maximum dans fifo MEDIUMQ de TaskComputing: 89
14 - Nb de paquets maximum dans fifo LOWQ de TaskComputing: 86
15 - Nb de paquets maximum dans port0 : 0
16 - Nb de paquets maximum dans port1 : 0
17 - Nb de paquets maximum dans port2 : 0

18- Message free : 6000
19- Message used : 0
20- Message used max : 251
21- Nombre de ticks depuis le début de l'execution 1687091
22- Pire temps video '0.0060047260'
23- Pire temps audio '0.0105250804'
24- Pire temps autre '0.0149534028'
----- Task stop suspend all tasks -----
----- Flags: 0 -----
```

Figure 3 : Manipulation 3

c) À partir des 3 figures obtenues en b), l'utilisation moyenne des 4 fifos (lignes 11 à 14) dépassent-elle la limite ?

Non, dans les trois manipulations, l'utilisation moyenne des 4 fifos (lignes 11 à 14) **ne dépasse pas la limite de 1024** qui est la taille maximum pour les fifos (fifo de queueing et les 3 fifos de taskComputer)

Preuve de capacité maximal pour taskQueueing

```
OSTaskCreate(&TaskQueueingTCB, "TaskQueueing", TaskQueueing, (void*)0,
TaskQueueingPRIO, &TaskQueueingSTK[0u], TASK_STK_SIZE / 2, TASK_STK_SIZE, 1024,
0, (void*) 0, (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR), &err );
```

Preuve de capacité maximal d'un fifo de taskComputing:

```
OSTaskCreate(&TaskComputingTCB[i], "TaskComputing", TaskComputing,  
&FIFO[i], TaskComputingPRIO+i, &TaskComputingSTK[i][0u], TASK_STK_SIZE / 2,  
TASK_STK_SIZE, 1024, 0, (void*) 0, (OS_OPT_TASK_STK_CHK |  
OS_OPT_TASK_STK_CLR), &err );
```

d) À partir des 3 figures obtenues en b), est-ce que le temps maximum pour traiter un paquet à travers les 3 fifos de TaskComputing HighQ, MediumQ et LowQ respectant les priorités demandées ?

- Manipulation 1 (Fréquence: 1000 Hz):

- Temps maximum de traitement pour HighQ: 0.091484521 s
- Temps maximum de traitement pour MediumQ: 0.182503300 s
- Temps maximum de traitement pour LowQ: 0.448507121 s

Analyse:

Dans cette première manipulation, les délais sont relativement élevés pour MediumQ et encore plus pour LowQ, ce qui est conforme aux priorités assignées. HighQ a le temps de traitement le plus rapide, suivi de MediumQ et enfin de LowQ, respectant l'ordre de priorité.

- Manipulation 2 (Fréquence: 8000 Hz, avec délai 8 fois plus long):

- Temps maximum de traitement pour HighQ: 0.0922041994 s
- Temps maximum de traitement pour MediumQ: 0.1887378967 s
- Temps maximum de traitement pour LowQ: 0.4495934546 s

Analyse:

Malgré la fréquence huit fois plus rapide (8000 Hz), les délais restent élevés. Les délais pour HighQ, MediumQ, et LowQ restent similaires à ceux de la manipulation 1. Cela signifie que les ajustements dans la fréquence du système n'ont pas eu un impact significatif sur les temps de traitement. Les délais observés pour MediumQ et LowQ sont toujours plus élevés, respectant les priorités.

- Manipulation 3 (Fréquence: 8000 Hz, mais traitement accéléré):

- Temps maximum de traitement pour HighQ: 0.0060047260 s
- Temps maximum de traitement pour MediumQ: 0.0105250804 s
- Temps maximum de traitement pour LowQ: 0.0149543028 s

Analyse:

Dans cette manipulation, le traitement est accéléré, et on observe une nette réduction des temps de traitement par rapport aux deux premières manipulations. HighQ, MediumQ et LowQ ont des délais beaucoup plus faibles tout en respectant les priorités. Le délai le plus court est pour HighQ, suivi de MediumQ et enfin de LowQ.

Conclusion:

Dans toutes les manipulations, les priorités sont bien respectées, et HighQ est toujours traité plus rapidement que MediumQ, qui est à son tour plus rapide que LowQ. Cependant, la manipulation 3 montre une amélioration notable des temps de traitement grâce à l'accélération du traitement.

e) À partir des 3 figures obtenues en b), est-ce que le temps maximum pour traiter un paquet à travers les 3 fifos de TaskComputing HighQ, MediumQ et LowQ respectant les priorités demandées ?

question répondue dans la réponse d!

f) Qu'observez-vous comme similitude/différence entre les 3 manipulations au niveau des stats de pire temps d'un paquet (no 22, 23 et 24 des statistiques) et justifiez au besoin.(mis à jours par le prof sur discord)

Pire temps d'un paquet vidéo (Stat no 22)

- Manipulation 1 : 0.0191488527
- Manipulation 2 : 0.0190824179
- Manipulation 3 : 0.0190146852

Observation : Les valeurs sont relativement similaires, et il n'y a pas de grandes variations entre les manipulations. Cela signifie que, pour la tâche vidéo, le traitement est resté globalement constant malgré les différentes fréquences et ajustements des paramètres.

Pire temps d'un paquet audio (Stat no 23)

- Manipulation 1 : 0.0188827307
- Manipulation 2 : 0.0188373967
- Manipulation 3 : 0.0181052804

Observation : Là aussi, les différences sont assez légères. Toutefois, on peut noter que le pire temps dans la manipulation 3 est légèrement plus bas comparé aux deux autres.

Pire temps d'un paquet "autre" (Stat no 24)

- Manipulation 1 : 0.0485807211
- Manipulation 2 : 0.0483954346
- Manipulation 3 : 0.0149534028

Observation : Il y a une nette différence dans le pire temps des paquets "autres" entre la manipulation 3 et les deux premières manipulations. La manipulation 3 affiche une réduction importante du pire temps, passant de valeurs autour de 0.048 à environ 0.015.

Question 3 Présentez votre capture d'écran de l'exécution de la valeur x trouvée ainsi que celle pour $x-1$. (Yacine)

Résultat pour $x=5ms$:

1. Vérification des dépassements des fifo :

D'après les résultats affichés (figure 5ms) :

- Queueing fifo : Maximum = 253
- HighQ fifo : Maximum = 92
- MediumQ fifo : Maximum = 231
- LowQ fifo : Maximum = 538

Aucun des fifos n'a atteint sa capacité maximale de 1024, donc **aucun dépassement** n'est observé.

Conclusion pour la première condition : La première condition est **respectée** car aucun fifo n'a débordé.

2. Vérification de la moyenne (inférieure à 50 % de la capacité, soit 512 messages) :

Calculons le pourcentage d'utilisation pour chaque fifo :

- Queueing fifo : 24.7%
- HighQ fifo : 8.98%
- MediumQ fifo : 22.56%
- LowQ fifo : 52.54%

Le fifo LowQ dépasse légèrement la limite de 50 %, avec une occupation à 52.54 %.

Conclusion :

Pour un délai de **5 ms** :

- La **première condition est respectée** : aucun fifo n'a débordé.
- La **deuxième condition est presque respectée**, mais le fifo **LowQ** dépasse légèrement la limite des **50 %** d'occupation (avec **52.54 %**).

```
----- Affichage des statistiques -----
Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 5
Frequence du systeme: 8000
1 - Nb de packets total crees : 384625
2 - Nb de packets total traitees : 344967
3 - Nb de packets rejetes pour mauvaise source : 35528
4 - Nb de packets rejetes pour mauvaise source total: 97
5 - Nb de packets rejetes pour mauvais CRC : 3838
6 - Nb de packets rejetes pour mauvais CRC total : 12
7 - Nb de packets rejetes fifo : 0
8 - Nb de packets rejetes fifo total : 0
9 - Nb de packets rejetes mauvaise priorites : 0
10 - Nb de packets rejetes mauvaise priorites total : 0
11 - Nb de packets maximum dans le fifo de Queueing : 253
12 - Nb de packets maximum dans le fifo HIGHQ de TaskComputing: 92
13 - Nb de packets maximum dans fifo MEDIUMQ de TaskComputing: 231
14 - Nb de packets maximum dans fifo LOWQ de TaskComputing: 538
15 - Nb de packets maximum dans port0 : 0
16 - Nb de packets maximum dans port1 : 0
17 - Nb de packets maximum dans port2 : 0

18- Message free : 5728
19- Message used : 272
20- Message used max : 681
21- Nombre de ticks depuis le debut de l'execution 241841
22- Pire temps video '0.0065916460'
23- Pire temps audio '0.0208573937'
24- Pire temps autre '0.0814012066'
----- Task stop suspend all tasks -----
----- Flags: 0 -----
```

Figure 4 : Résultat pour Delai pour vider les fifos msec: 5ms

Résultat pour x-1= 4ms:

1. Vérification des dépassements des fifo :

- **Queueing fifo** : Maximum = 253
- **HighQ fifo** : Maximum = 93
- **MediumQ fifo** : Maximum = 311
- **LowQ fifo** : Maximum = **1024**

Le fifo **LowQ** a atteint **1024**, ce qui signifie qu'il a **débordé**.

Conclusion pour la première condition : La première condition n'est **pas respectée** pour un délai de 4 ms, car le fifo LowQ a atteint sa capacité maximale, ce qui a causé des rejets.

2. Vérification de la moyenne (inférieure à 50 % de la capacité, soit 512 messages) :

Calculons le pourcentage d'utilisation pour chaque fifo :

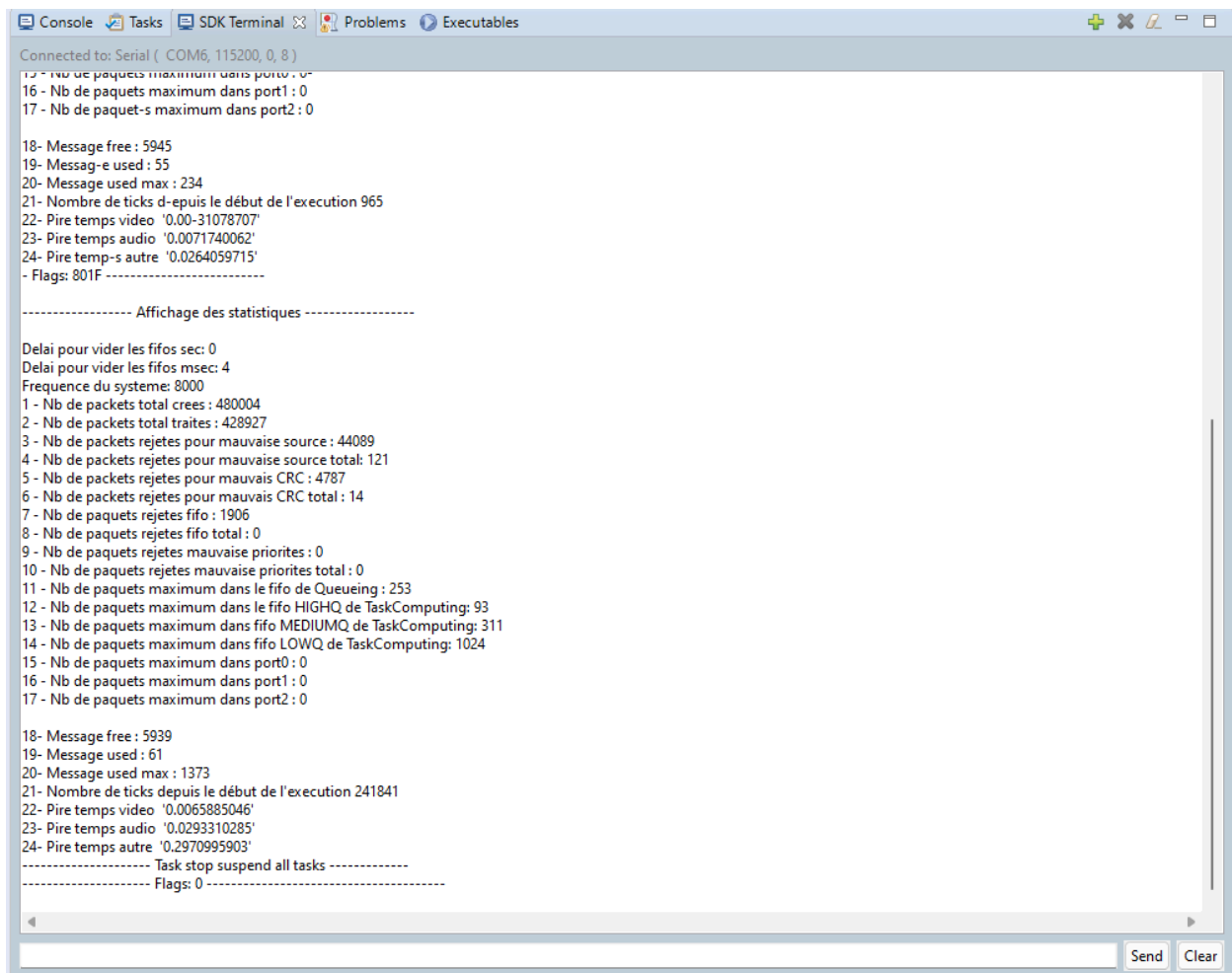
- **Queueing fifo** : 24.7%
- **HighQ fifo** : 9.08%
- **MediumQ fifo** : 30.37%
- **LowQ fifo** : 100%

Le fifo **LowQ** a atteint **100 %**, ce qui signifie que la **deuxième condition n'est pas respectée** non plus, puisque l'utilisation moyenne dépasse largement les **50 %** de la capacité maximale.

Conclusion :

Pour un délai de **4 ms**, les **deux conditions ne sont pas respectées** :

- Le fifo **LowQ** a atteint sa capacité maximale de **1024**, ce qui a entraîné des rejets.
- L'occupation du fifo **LowQ** est à **100 %**, ce qui dépasse les **50 %** requis.



```
Connected to: Serial ( COM6, 115200, 0, 8 )
12- Nb de paquets maximum dans port0 : 0
16- Nb de paquets maximum dans port1 : 0
17- Nb de paquet-s maximum dans port2 : 0

18- Message free : 5945
19- Message used : 55
20- Message used max : 234
21- Nombre de ticks d-eups le début de l'exécution 965
22- Pire temps video '0.00-31078707'
23- Pire temps audio '0.0071740062'
24- Pire temp-s autre '0.0264059715'
- Flags: 801F -----

----- Affichage des statistiques -----

Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 4
Frequence du systeme: 8000
1 - Nb de packets total crees : 480004
2 - Nb de packets total traites : 428927
3 - Nb de packets rejetees pour mauvaise source : 44089
4 - Nb de packets rejetees pour mauvaise source total: 121
5 - Nb de packets rejetees pour mauvais CRC : 4787
6 - Nb de packets rejetees pour mauvais CRC total : 14
7 - Nb de paquets rejetees fifo : 1906
8 - Nb de paquets rejetees fifo total : 0
9 - Nb de paquets rejetees mauvaise priorites : 0
10 - Nb de paquets rejetees mauvaise priorites total : 0
11 - Nb de paquets maximum dans le fifo de Queueing : 253
12 - Nb de paquets maximum dans le fifo HIGHQ de TaskComputing: 93
13 - Nb de paquets maximum dans fifo MEDIUMQ de TaskComputing: 311
14 - Nb de paquets maximum dans fifo LOWQ de TaskComputing: 1024
15 - Nb de paquets maximum dans port0 : 0
16 - Nb de paquets maximum dans port1 : 0
17 - Nb de paquets maximum dans port2 : 0

18- Message free : 5939
19- Message used : 61
20- Message used max : 1373
21- Nombre de ticks depuis le début de l'exécution 241841
22- Pire temps video '0.0065885046'
23- Pire temps audio '0.0293310285'
24- Pire temps autre '0.2970995903'
----- Task stop suspend all tasks -----
----- Flags: 0 -----
```

Figure 5 : Résultat pour Delai pour vider les fifos msec: 4ms

Question 4

a) Affichez d'abord les performances obtenus : capture d'écran de la valeur x trouvé à la question 3 (qui sera la solution sans section critique) + 2 les captures d'écran de la section 5.2.3.2 (i.e. avec section critique pour HIGHQ et LOWQ protégé par mutex, puis avec section critique pour HIGHQ et LOWQ protégé par sémaphore).

```
----- Affichage des statistiques -----
Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 5
Frequence du systeme: 8000
1 - Nb de packets total crees : 384625
2 - Nb de packets total traites : 344967
3 - Nb de packets rejetes pour mauvaise source : 35528
4 - Nb de packets rejetes pour mauvaise source total: 97
5 - Nb de packets rejetes pour mauvais CRC : 3838
6 - Nb de packets rejetes pour mauvais CRC total : 12
7 - Nb de paquets rejetes fifo : 0
8 - Nb de paquets rejetes fifo total : 0
9 - Nb de paquets rejetes mauvaise priorites : 0
10 - Nb de paquets rejetes mauvaise priorites total : 0
11 - Nb de paquets maximum dans le fifo de Queueing : 253
12 - Nb de paquets maximum dans le fifo HIGHQ de TaskComputing: 92
13 - Nb de paquets maximum dans fifo MEDIUMQ de TaskComputing: 231
14 - Nb de paquets maximum dans fifo LOWQ de TaskComputing: 538
15 - Nb de paquets maximum dans port0 : 0
16 - Nb de paquets maximum dans port1 : 0
17 - Nb de paquets maximum dans port2 : 0

18- Message free : 5728
19- Message used : 272
20- Message used max : 681
21- Nombre de ticks depuis le début de l'execution 241841
22- Pire temps video '0.0065916460'
23- Pire temps audio '0.0208573937'
24- Pire temps autre '0.0814012066'
----- Task stop suspend all tasks -----
----- Flags: 0 -----
```

Figure 6 : Valeur x trouvé à la question 3

```
----- Affichage des statistiques -----
Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 5
Frequence du systeme: 8000
1 - Nb de packets total crees : 384948
2 - Nb de packets total traites : 345166
3 - Nb de packets rejetes pour mauvaise source : 35535
4 - Nb de packets rejetes pour mauvaise source total: 97
5 - Nb de packets rejetes pour mauvais CRC : 3841
6 - Nb de packets rejetes pour mauvais CRC total : 12
7 - Nb de paquets rejetes fifo : 0
8 - Nb de paquets rejetes fifo total : 0
9 - Nb de paquets rejetes mauvaise priorites : 0
10 - Nb de paquets rejetes mauvaise priorites total : 0
11 - Nb de paquets maximum dans le fifo de Queueing : 253
12 - Nb de paquets maximum dans le fifo HIGHQ de TaskComputing: 92
13 - Nb de paquets maximum dans fifo MEDIUMQ de TaskComputing: 233
14 - Nb de paquets maximum dans fifo LOWQ de TaskComputing: 539
15 - Nb de paquets maximum dans port0 : 0
16 - Nb de paquets maximum dans port1 : 0
17 - Nb de paquets maximum dans port2 : 0

18- Message free : 5628
19- Message used : 372
20- Message used max : 683
21- Nombre de ticks depuis le début de l'execution 241841
22- Pire temps video '0.0065982309'
23- Pire temps audio '0.0211261939'
24- Pire temps autre '0.0815311149'
----- Task stop suspend all tasks -----
----- Flags: 0 -----
```

Figure 7 : utilisation du mutex pour la section critique HighQ et LowQ

```
----- Affichage des statistiques -----
Delai pour vider les fifos sec: 0
Delai pour vider les fifos msec: 5
Frequence du systeme: 8000
1 - Nb de packets total crees : 384948
2 - Nb de packets total traites : 345168
3 - Nb de packets rejetes pour mauvaise source : 35536
4 - Nb de packets rejetes pour mauvaise source total: 97
5 - Nb de packets rejetes pour mauvais CRC : 3841
6 - Nb de packets rejetes pour mauvais CRC total : 12
7 - Nb de paquets rejetes fifo : 0
8 - Nb de paquets rejetes fifo total : 0
9 - Nb de paquets rejetes mauvaise priorites : 0
10 - Nb de paquets rejetes mauvaise priorites total : 0
11 - Nb de paquets maximum dans le fifo de Queueing : 253
12 - Nb de paquets maximum dans le fifo HIGHQ de TaskComputing: 166
13 - Nb de paquets maximum dans fifo MEDIUMQ de TaskComputing: 233
14 - Nb de paquets maximum dans fifo LOWQ de TaskComputing: 541
15 - Nb de paquets maximum dans port0 : 0
16 - Nb de paquets maximum dans port1 : 0
17 - Nb de paquets maximum dans port2 : 0

18- Message free : 5627
19- Message used : 373
20- Message used max : 683
21- Nombre de ticks depuis le début de l'execution 241841
22- Pire temps video '0.0153257381'
23- Pire temps audio '0.0211249348'
24- Pire temps autre '0.0854026005'
----- Task stop suspend all tasks -----
----- Flags: 0 -----
```

Figure 8 : utilisation de sémaphore pour la section critique de HighQ et LowQ

b) Comparez les 3 captures d'écran de a) en focusant sur le temps maximum d'un paquet HIGHQ. Que peut-on conclure?

Lorsqu'on compare les trois captures d'écran (sans section critique, avec mutex, et avec sémaphore), on se concentre sur le **temps maximum de traitement d'un paquet HIGHQ**. Voici ce que nous pouvons observer et conclure :

L'utilisation du mutex ajoute un léger surcoût mais permet de maintenir une bonne gestion des priorités grâce à l'héritage de priorité. En revanche, l'utilisation d'un sémaphore semble avoir introduit un délai plus important, probablement en raison de la gestion différente du sémaphore, qui ne garantit pas l'héritage de priorité comme le mutex. Cela montre que le mutex est plus adapté pour des sections critiques nécessitant une gestion stricte des priorités dans un environnement en temps réel.

c) Calculez le blocage maximum de la tâche HIGHQ de TaskComputing. Expliquez bien vos calculs.

Le blocage maximum pour une tâche est le temps qu'une tâche avec une priorité élevée, comme HIGHQ, pourrait être bloquée par des tâches de plus basse priorité qui accèdent à la même section critique.

Avec Mutex ou Semaphore: Le blocage maximum est limité à la durée pendant laquelle une tâche de priorité inférieure détient le mutex ou le semaphore. Dans ce cas, la tâche LOWQ peut provoquer un blocage maximum correspondant à son temps d'exécution dans la section critique (traitement d'un paquet, calcul du délai d'attente).

Temps d'attente aléatoire pour LOWQ : [0-2 ticks]. Le blocage maximal pour HIGHQ est donc **2 ticks**, soit environ 0.00025 secondes (car 1 tick = 0.000125 s avec une fréquence de 8000 Hz).

d) Supposons qu'on augmente le délai aléatoire de MEDIUMQ à 3 ticks. Calculez le blocage maximum de la tâche HIGHQ de TaskComputing. Expliquez bien vos calculs.

Si la tâche MEDIUMQ a un délai aléatoire allant jusqu'à **3 ticks** et qu'elle est incluse dans la section critique, cela augmente le blocage possible pour HIGHQ. Le blocage maximum pour HIGHQ devient la somme des délais possibles de LOWQ et MEDIUMQ dans la section critique:

- LOWQ : [0-2 ticks]
- MEDIUMQ : [0-3 ticks]

Ainsi, le blocage maximum serait de **5 ticks** (2 + 3), soit **0.000625 secondes**.

e) Supposons que l'on remplace la gestion de la section critique par un fifo qui au départ contient 1 seul élément. Lorsqu'une tâche rentre dans cette section critique elle consomme cet élément et quand la même tâche quitte la section critique elle produit cet élément. c.1) Montrez qu'on a bel et bien un comportement capable de gérer une section critique. c.2) Sans faire

l'implémentation, à quel mécanisme (mutex ou sémaphore) devrait s'approcher les performances après plus de 100,000 paquets générés.

Démonstration du comportement du FIFO :

Lorsque l'on remplace la gestion de la section critique par un **FIFO** contenant un seul élément, on assure que :

1. Lorsqu'une tâche entre dans la section critique, elle consomme l'unique élément du FIFO.
2. Quand elle quitte la section critique, elle produit cet élément, permettant à une autre tâche d'y accéder.

Cela fonctionne exactement comme un mécanisme de **verrouillage** (comme un sémaphore) :

- Une tâche ne peut pas entrer dans la section critique tant qu'une autre tâche n'a pas terminé et rendu l'élément.
- Cela garantit que **seule une tâche** à la fois peut accéder à la section critique, comme avec un mutex ou un sémaphore.

Performances comparées après 100 000 paquets générés :

Après avoir généré plus de **100 000 paquets**, le comportement du FIFO devrait **s'approcher des performances d'un sémaphore**. En effet :

- Le FIFO permet un accès séquentiel et bloquant, tout comme un sémaphore comptant avec une seule ressource.
- Le **FIFO** n'a pas la flexibilité du mutex en termes de priorisation des tâches, mais en termes de gestion de l'accès à une section critique, son comportement est proche de celui d'un sémaphore.

Ainsi, les performances à long terme ressembleraient davantage à celles d'un système utilisant un **sémaphore** qu'à celles utilisant un **mutex**, en raison du contrôle strict de l'accès au FIFO et de l'ordre séquentiel imposé.

Question 5 Expliquez comment se fait la gestion de mémoire de uC/OS-III et quels sont les avantages de cette gestion par rapport à des appels de malloc et free, quand on fait du temps réel.

uC/OS-III utilise une gestion de mémoire basée sur des **pools de mémoire statiques**, où les blocs de mémoire sont prédéfinis lors de l'initialisation du système. Chaque pool contient un nombre fixe de blocs de taille constante qui peuvent être alloués et libérés pendant l'exécution. Contrairement aux fonctions **malloc** et **free** qui allouent dynamiquement de la mémoire à l'exécution, cette approche garantit une **allocation déterministe** et rapide, car le temps nécessaire pour allouer ou libérer un bloc de mémoire reste constant ($O(1)$).

Cette gestion est particulièrement avantageuse dans les systèmes temps réel pour plusieurs raisons :

1. **Temps d'allocation prédictible** : L'allocation et la libération se font en temps constant, ce qui permet de respecter les contraintes de temps critique imposées par les tâches en temps réel. À l'inverse, **malloc** et **free** peuvent introduire des délais imprévisibles dus à la gestion dynamique de la mémoire.
2. **Réduction de la fragmentation** : **malloc** peut causer de la fragmentation mémoire, créant des blocs inutilisables à long terme. Les pools de mémoire de uC/OS-III évitent ce problème en ayant des blocs de taille fixe, ce qui assure une meilleure utilisation de la mémoire et élimine les risques de fragmentation interne ou externe.
3. **Fiabilité accrue** : En utilisant des blocs de mémoire prédéfinis, le système évite les erreurs courantes liées à l'allocation dynamique, telles que les fuites de mémoire. Cela garantit que les tâches critiques auront toujours accès à la mémoire nécessaire, tant que des blocs sont disponibles dans le pool.