Systèmes embarqués temps réel (GIF-3004)
Département de génie électrique et de génie informatique
Hiver 2017

UNIVERSITÉ LAVAL

EXAMEN FINAL

Instructions : – Une feuille aide-mémoire recto verso manuscrite est permise ;
– Durée de l'examen : 1 h 50.

Pondération : Cet examen compte pour 25% de la note finale.

Aide-mémoire

Combinatoire

- Factorielle de n : $n! = 1 \times 2 \times 3 \times \cdots \times n \qquad (0! = 1)$
- Choisir k parmi n, selon un certain ordre (arrangement) : $A_n^k = \frac{n!}{(n-k)!}$
- Choisir k parmi n, sans ordre (combinaison) : $\binom{n}{k} = \frac{A_n^k}{k!} = \frac{n!}{k!(n-k)!}$

Loi exponentielle

- Densité de probabilité : $p(t) = \lambda \exp(-\lambda t)$, pour $t \geq 0$
- Loi de répartition : $P(X \leq t) = 1 - \exp(-\lambda t)$, pour $t \geq 0$
- Espérance : $\mathbb{E}(t) = \frac{1}{\lambda}$
- Variance : $\mathrm{Var}(t) = \frac{1}{\lambda^2}$

Théorie des files d'attente

Modèle	$M/M/1$	$M/M/m$	$M/M/\infty$	$M/M/1/K$
Utilisation (ρ)	$\frac{\lambda}{\mu}$	$\frac{\lambda}{m\mu}$	$\frac{\lambda}{\mu}$	$\frac{\lambda}{\mu}$
Probabilité d'aucune tâche (π_0)	$1 - \frac{\lambda}{\mu} = 1 - \rho$	$\left[1 + \sum_{n=1}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!} \frac{1}{1-\rho}\right]^{-1}$	$\exp(-\rho)$	$\left[1 + \sum_{n=1}^{K} (\lambda/\mu)^n\right]^{-1} = \frac{1-\rho}{1-\rho^{K+1}}$
Probabilité de n tâches (π_n)	$(1-\rho)\rho^n$	$\pi_0 \frac{(m\rho)^n}{n!}$ pour $1 \leq n \leq m$ $\pi_0 \frac{m^m}{m!} \rho^n$ pour $n > m$	$\exp(-\rho) \frac{\rho^n}{n!}$	$\frac{1-\rho}{1-\rho^{K+1}} \rho^n$ pour $0 \leq n \leq K$ 0 pour $n > K$
Taille moyenne de la file ($\mathbb{E}(X)$)	$\frac{\rho}{1-\rho}$	$\sum_{n=0}^{\infty} n\pi_n = m\rho + \frac{(m\rho)^m}{m!} \frac{\rho}{(1-\rho)^2} \pi_0$	ρ	$\frac{\rho}{1-\rho^{K+1}} \left[\frac{1-\rho^K}{1-\rho} - K\rho^K\right]$
Temps système moyen ($\mathbb{E}(S)$)	$\frac{1/\mu}{1-\rho}$	$\frac{1}{\mu} + \frac{1}{\mu} \frac{(m\rho)^m}{m!} \frac{\pi_0}{m(1-\rho)^2}$	$\frac{1}{\mu}$	$\frac{\mathbb{E}(X)}{\lambda(1-\pi_K)}$

Question 1 (20 points sur 100)

Supposons que vous travaillez à la conception d'un système de conduite de véhicules autonomes. Le système doit effectuer un traitement en temps réel, en traitant l'information reçue par les différents capteurs du véhicule et prenant des décisions de conduite à intervalle régulier.

(8) (a) Supposons que le système peut effectuer une certaine charge de travail sur un seul processeur en 250 ms. 75 % de ce traitement peut être parfaitement parallélisé, les éléments restants du traitement devant être exécutés sur un seul processeur. Déterminez le nombre de processeurs que l'on doit prévoir pour que le système puisse effectuer un traitement en temps réel pour chaque charge de travail en 100 ms ou moins.

Solution: Nous allons utiliser la loi d'Amdahl pour résoudre ce problème :

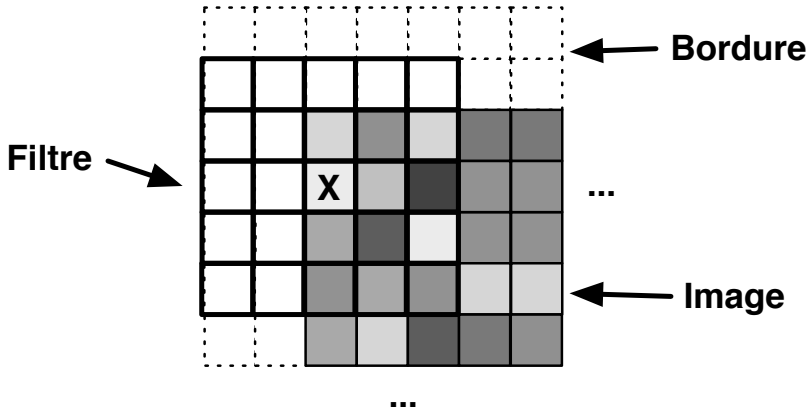
$$S_p = \frac{1}{f + \frac{1-f}{p}} = \frac{1}{0{,}25 + \frac{0{,}75}{p}} = \frac{250\,\text{ms}}{100\,\text{ms}} = 2{,}5$$

$$1 = 2{,}5 \times 0{,}25 + \frac{2{,}5 \times 0{,}75}{p} = 0{,}625 + \frac{1{,}875}{p}$$

$$p = \frac{1{,}875}{1 - 0{,}625} = \frac{1{,}875}{0{,}375} = 5$$

Cinq processeurs sont donc nécessaires pour effectuer le traitement d'une charge de travail en 100 ms.

(12) (b) Supposons maintenant que l'on doit traiter sur un GPU des images en tons de gris de 256×256 pixels provenant d'une caméra de la voiture. Le traitement consiste en la convolution d'un filtre de 5×5, qui s'applique à chaque pixel de l'image. L'application du filtre sur des pixels en bordure de l'image implique de définir une bordure de deux pixels de large de chaque côté de l'image tel qu'illustré ici-bas, où le filtre est appliqué au pixel de la deuxième ligne et première colonne de l'image.



On veut paralléliser les opérations en découpant l'image en sous-images de plus petite taille, qui seront traitées en parallèle sur les unités de traitement du GPU. Pour chaque image, une bordure de deux pixels de large des quatre côtés de l'image doit être utilisée,

en utilisant les pixels voisins de la sous-image lorsque la bordure de celle-ci ne correspond pas à la bordure de l'image. L'application du filtre sur un pixel implique 25 multiplications, 24 additions et une écriture, donc 50 opérations au total. Le coût du chargement d'une sous-image dans la mémoire d'une unité de traitement correspond à une opération par pixel. On doit inclure dans ce calcul le coût de chargement des pixels correspondant aux bordures supplémentaires et aux pixels pertinents des sous-images voisines. On suppose que l'opération de chargement en mémoire des unités de traitement des sous-images doit être effectuée **séquentiellement** sur le GPU.

Déterminez l'équation générale correspondant au temps de traitement parallèle d'un tel modèle, en nombre d'opérations.

Déterminez également la configuration la plus efficace en termes de temps de traitement parallèle selon les différentes configurations suivantes :

— 16 unités de traitement traitant des sous-images de 64×64 pixels ;
— 64 unités de traitement traitant des sous-images de 32×32 pixels ;
— 256 unités de traitement traitant des sous-images de 16×16 pixels ;
— 1024 unités de traitement traitant des sous-images de 8×8 pixels.

Justifiez votre réponse avec calculs.

Solution: L'équation générale du temps de traitement parallèle est :

$$T_p = k_{\text{conv}} \times \frac{n_{\text{pixels}}}{p} + k_{\text{mem}} \times n_{\text{subimg}} \times p$$

où :

— k_{conv} : nombre d'opérations d'une convolution sur un pixel (ici 50 opérations) ;
— n_{pixels} : nombre de pixels d'une image complète ;
— k_{mem} : nombre d'opérations pour charger un pixel en mémoire d'une unité de traitement (ici 1 opération) ;
— n_{subimg} : nombre de pixels des sous-images, incluant bordure et pixels pertinents de sous-images voisines, chargés sur un unité de traitement ;
— p : nombre de processeurs utilisés.

Temps parallèle pour les différentes configurations testées :

— 16 unités de traitement traitant des sous-images de 64×64 pixels :

$$T_{16} = 50 \times \frac{256 \times 256}{16} + 1 \times (64+4)^2 \times 16 = 204\,800 + 73\,984 = 278\,784$$

— 64 unités de traitement traitant des sous-images de 32×32 pixels :

$$T_{64} = 50 \times \frac{256 \times 256}{64} + 1 \times (32+4)^2 \times 64 = 51\,200 + 82\,944 = 134\,144$$

— 256 unités de traitement traitant des sous-images de 16×16 pixels :

$$T_{256} = 50 \times \frac{256 \times 256}{256} + 1 \times (16+4)^2 \times 256 = 12\,800 + 102\,400 = 115\,200$$

— 1024 unités de traitement traitant des sous-images de 8×8 pixels :

$$T_{1024} = 50 \times \frac{256 \times 256}{1024} + 1 \times (8+4)^2 \times 1024 = 3\,200 + 147\,456 = 150\,656$$

La configuration à 256 unités de traitement est donc la plus performante en termes de nombre d'opérations effectuées.

Question 2 (20 points sur 100)

Supposons maintenant que le véhicule autonome inclut un système de vision numérique capable de faire le suivi des déplacements d'entités d'intérêt. Ces entités se situent dans l'environnement immédiat autour de la voiture, par exemple d'autres véhicules, des piétons ou des cyclistes. Le système doit faire le suivi de chaque entité d'intérêt visible par les capteurs de la voiture. Une analyse du nombre d'entités à suivre a été effectuée par un modèle de files d'attente, où il a été déterminé qu'une nouvelle entité d'intérêt devient visible aux capteurs en moyenne à chaque 10 s et reste visible pour le suivi pendant 4 s, en moyenne. Cette modélisation fait l'hypothèse que les arrivées et les départs suivent des processus aléatoires de Poisson, sans mémoire. Le système est conçu pour pouvoir suivre simultanément jusqu'à quatre entités d'intérêt.

(14) (a) Calculez la probabilité instantanée, en régime permanent, que le nombre d'entités pour lesquelles le suivi doit être effectué dépasse la limite du système, soit de pouvoir suivre au maximum quatre entités simultanément.

Solution: Ce modèle correspond à une file d'attente de type $M/M/m$. Selon les données du problème, on sait donc que les arrivées se font à un taux $\lambda = 1/\mathbb{E}(Y) = 1/10 = 0{,}1$, alors que le taux de départ est de $\mu = 1/\mathbb{E}(Z) = 1/4 = 0{,}25$.

Pour répondre à la question, nous pouvons utiliser les équations de files d'attente de type $M/M/m$ données dans l'aide-mémoire de la première page :

$$\rho = \frac{\lambda}{m\mu}$$

$$\pi_0 = \left[1 + \sum_{n=1}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!}\frac{1}{1-\rho}\right]^{-1}$$

$$\pi_n = \pi_0 \frac{(m\rho)^n}{n!} \quad \text{pour } n = 1,2,\ldots,m$$

$$P(X > m) = \sum_{n=m+1}^{\infty} \pi_n = 1 - \sum_{n=0}^{m} \pi_n$$

Selon les données de la question :

$$
\begin{aligned}
\rho &= \frac{\lambda}{m\mu} = \frac{0{,}1}{4 \times 0{,}25} = 0{,}1 \\
\pi_0 &= \left[1 + \frac{(4 \times 0{,}1)^1}{1!} + \frac{(4 \times 0{,}1)^2}{2!} + \frac{(4 \times 0{,}1)^3}{3!} + \frac{(4 \times 0{,}1)^4}{4!}\frac{1}{1-0{,}1}\right]^{-1} \\
&= \left[1 + 0{,}4 + \frac{0{,}16}{2} + \frac{0{,}064}{6} + \frac{0{,}0256}{24}\frac{1}{0{,}9}\right]^{-1} \\
&= \left[1 + 0{,}4 + 0{,}08 + 0{,}01067 + \frac{0{,}001067}{0{,}9}\right]^{-1} \\
&= [1 + 0{,}4 + 0{,}08 + 0{,}01067 + 0{,}001185]^{-1} = 1/1{,}4919 = 0{,}6703 \\
\pi_1 &= 0{,}6703\frac{(4 \times 0{,}1)^1}{1!} = 0{,}6703 \times 0{,}4 = 0{,}2681 \\
\pi_2 &= 0{,}6703\frac{(4 \times 0{,}1)^2}{2!} = 0{,}6703 \times \frac{0{,}16}{2} = 0{,}05362 \\
\pi_3 &= 0{,}6703\frac{(4 \times 0{,}1)^3}{3!} = 0{,}6703 \times \frac{0{,}064}{6} = 0{,}007150 \\
\pi_4 &= 0{,}6703\frac{(4 \times 0{,}1)^4}{4!} = 0{,}6703 \times \frac{0{,}0256}{24} = 0{,}0007150 \\
P(X > m) &= 1 - (\pi_0 + \pi_1 + \pi_2 + \pi_3 + \pi_4) \\
&= 1 - (0{,}6703 + 0{,}2681 + 0{,}05362 + 0{,}00715 + 0{,}000715) \\
&= 1 - 0{,}999885 = 0{,}000115 \\
&\approx 0{,}00012
\end{aligned}
$$

Donc, des entités d'intérêt ne pourront pas être suivies environ 0,012 % du temps.

(6) (b) Est-ce que les hypothèses faites pour la modélisation par file d'attente des entités d'intérêt devant être suivies apparaissent raisonnables en pratique, pour ce problème de suivi ? Justifiez de façon convaincante votre réponse.

Solution: Non, en pratique les résultats devraient différer significativement de cette modélisation. En effet, supposer l'utilisation de processus de Poisson fait l'hypothèse d'indépendance entre les arrivées dans le système. Cependant, en pratique, on sait que les entités d'intérêt devant être suivies par le véhicule risquent d'arriver groupées, par exemple lorsque plusieurs véhicules sont ralentis lorsque le traffic est lourd ou lorsque certaines rues sont particulièrement bondées de piétons. Un processus va probablement sous-estimer la fréquence de ces situations où de nombreuses entités doivent être suivies simultanément.

Question 3 (20 points sur 100)

Supposons maintenant que l'on s'intéresse à la fiabilité de composantes du véhicule autonome.

(10) (a) Supposons que la voiture comporte un dispositif de trois capteurs permettant de mesurer la distance d'objets externes. On suppose que la fiabilité individuelle des capteurs de la matrice est identique pour tous les capteurs, soit une durée de vie suivant une loi exponentielle avec un temps espéré (moyen) de fonctionnement de 100 000 heures. De plus, on fait l'hypothèse que les défaillances des capteurs sont indépendantes et que les mesures de distance sont utilisables tant qu'au moins deux capteurs sont fonctionnels. Donnez la probabilité de défaillance du dispositif ne soit plus en mesure de retourner des mesures de distance utilisables au temps t.

Solution: La paramétrisation de la loi exponentielle de la fiabilité d'un capteur est :

$$\mathbb{E}(T) = \frac{1}{\lambda} = 100\,000,$$
$$\lambda = \frac{1}{100\,000} = 10^{-5}.$$

La fiabilité d'un capteur est donnée par la loi de répartition :

$$F(t) = P(X \leq t) = 1 - \exp(-\lambda t),$$
$$R(t) = P(X > t) = 1 - F(t) = \exp(-\lambda t).$$

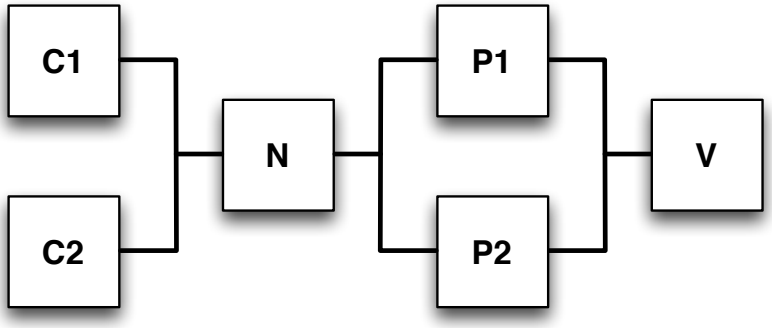
La fiabilité du système à l'effet qu'au moins k composants sur n soient fonctionnels au temps t est :

$$\begin{aligned} R_s(t) &= \sum_{i=k}^{n} \binom{n}{i} [R(t)]^i [1 - R(t)]^{n-i} \\ &= \sum_{i=k}^{n} \binom{n}{i} [\exp(-\lambda t)]^i [1 - \exp(-\lambda t)]^{n-i} \\ &= \sum_{i=k}^{n} \binom{n}{i} \exp(-\lambda\, i\, t) [1 - \exp(-\lambda t)]^{n-i}. \end{aligned}$$

Et pour le cas présent, avec $n = 3$, $k = 2$ et $\lambda = 10^{-5}$:

$$\begin{aligned} R_s(t) &= \sum_{i=2}^{3} \binom{3}{i} \exp(-10^{-5} \times i \times t)[1 - \exp(-10^{-5} \times t)]^{3-i} \\ &= \frac{3!}{2!\,1!} \exp(-10^{-5} \times 2 \times t)[1 - \exp(-10^{-5} \times t)]^1 \\ &\quad + \frac{3!}{3!\,0!} \exp(-10^{-5} \times 3 \times t)[1 - \exp(-10^{-5} \times t)]^0 \\ &= 3 \times \exp(-2 \times 10^{-5} \times t)[1 - \exp(-10^{-5} \times t)] + \exp(-3 \times 10^{-5} t) \\ &= 3 \times \exp(-2 \times 10^{-5} \times t) - 3 \times \exp(-3 \times 10^{-5} \times t) + \exp(-3 \times 10^{-5} \times t) \\ &= 3 \times \exp(-2 \times 10^{-5} \times t) - 2 \times \exp(-3 \times 10^{-5} \times t). \end{aligned}$$

(10) (b) Supposons maintenant la modélisation suivante d'un sous-système de la voiture.



Voici les caractéristiques des composants de ce sous-système :

— Les composants C1 et C2 sont des capteurs fonctionnant en parallèle – le système fonctionne tant qu'au moins un capteur fonctionne – dont la fiabilité individuelle est R_C ;

— Le composant N désigne l'interconnexion entre les capteurs C1 et C2 et les processeurs P1 et P2, dont la fiabilité est R_N ;

— Les composants P1 et P2 correspondent à deux processeurs redondants, dont la fiabilité individuelle est R_P ;

— Le composant V correspond au contrôle du véhicule selon les traitements faits sur les processeurs P1 et P2, dont la fiabilité est R_V.

Déterminez la fiabilité globale du sous-système modélisé ainsi.

Solution: Pour déterminer la fiabilité globale du sous-système, nous allons d'abord réduire les éléments en parallèle, soit les capteurs et les processeurs. Selon l'équation pour des systèmes parallèles, la réduction avec deux composants est :

$$R_{||} = 1 - \prod_i F_i = 1 - F_1\,F_2 = 1 - (1 - R_1)(1 - R_2)$$
$$= 1 - 1 + R_1 + R_2 - R_1\,R_2 = R_1 + R_2 - R_1\,R_2$$

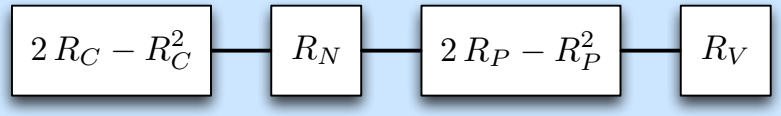
Pour les capteurs, la fiabilité correspondante est :

$$R_{CS} = 2\,R_C - R_C^2,$$

alors que pour les processeurs, la fiabilité est :

$$R_{PS} = 2\,R_P - R_P^2.$$

Donc, le système dont les composants parallèles sont réduits revient ce qui suit.



La fiabilité du sous-système est donc :

$$\begin{aligned} R_S &= \prod_i R_i \\ &= (2\,R_C - R_C^2) R_N (2\,R_P - R_P^2) R_V \\ &= 4\,R_C\,R_N\,R_P\,R_V - 2\,R_C^2\,R_N\,R_P\,R_V - 2\,R_C\,R_N\,R_P^2\,R_V + R_C^2\,R_N\,R_P^2\,R_V. \end{aligned}$$

Question 4 (40 points sur 100)

Répondez aussi brièvement et clairement que possible aux questions suivantes.

(4) (a) Expliquez la différence entre la moitié supérieure (*top-half*) et la moitié inférieure (*bottom-half*) d'un pilote dans le noyau Linux.

Solution: La moitié supérieure représente la portion d'un pilote qui effectue une exécution rapide dans le contexte de l'interruption, pour traitement minimal non préemptible. La moitié inférieure permet une exécution plus longue et préemptible dans le contexte courant (ex. kthread, tasklet), pour compléter les traitements complexes associés à l'interruption.

(4) (b) Présentez la différence entre la classe des dispositifs de caractères (*character devices*) et la classe des dispositifs de blocs (*block devices*) des périphériques du noyau Linux, ainsi que des exemples de dispositifs de ces deux classes.

Solution:

Dispositifs de caractères permettent un échange de données selon un flux séquentiel. Des périphériques comme des ports série où le clavier correspond à cette classe de dispositifs.

Dispositifs de blocs permettent des accès aléatoires à des morceaux de taille fixe. Les dispositifs de stockage massifs comme les disques durs et les clés USB de mémoire correspondent à ce modèle.

(4) (c) Indiquez où peut-on récupérer les messages écrits via l'appel à la fonction `printk` dans le noyau Linux.

Solution: Les messages peuvent être obtenus en appelant la commande `dmesg` dans le terminal. L'information est également inscrite dans le fichier `/var/log/syslog`. On peut même obtenir le résultat de `printk` sur un lien série, pour débogage du noyau.

(4) (d) Expliquez pourquoi les systèmes temps réel et la communication sur réseaux TCP/IP ne font pas bon ménage.

Solution: La communication sur réseaux TCP/IP comporte généralement une grande variabilité dans les délais d'acheminement des messages. Cette variabilité est induite par différents facteurs, tels l'engorgement du réseau et le réacheminement en cas de corruption du message. Dans un contexte de systèmes temps réel, on veut éliminer ces variabilités pour garantir que l'exécution des tâches du système se fasse selon les contraintes de l'application.

(4) (e) Expliquez à quoi sert le protocole NTP (*Network Time Protocol*).

Solution: Le protocole NTP permet d'assurer la cohérence du temps entre plusieurs serveurs. Cet ajustement se fait en moyennant des ajustements graduels du décalage, par des échanges bidirectionnels des valeurs du temps entre plusieurs serveurs.

(4) (f) Expliquez la différence entre le délai (latence) et la variabilité (*jitter*) dans un contexte de communication avec qualité de service (*QoS*).

Solution: Le délai correspond au temps entre le moment où un message est envoyé par l'expéditeur et le moment où le message est reçu par le destinataire. La variabilité correspond à des situations où les délais pourraient varier significativement et aléatoirement. Un lien de communication peut fonctionner avec délai moyen élevé tout en ayant une faible variabilité, et vice-versa.

(4) (g) Dans la notation $A/B/m/K$ des files d'attente, indiquez à quoi correspond le K.

Solution: La valeur de K correspond à la longueur de la file d'attente, c'est-à-dire le nombre de requêtes pouvant être conservé dans le système incluant les requêtes présentement en service.

(4) (h) Indiquez le principal avantage pour le programmeur de travailler avec un système multiprocesseur à **mémoire partagée**.

Solution: Avec un système multiprocesseur à mémoire partagée, le programme à accès à l'ensemble de la mémoire sur tous les cœurs de traitement. Ainsi, la synchronisation et la communication entre les cœurs de traitement peut se faire via la mémoire partagée, en utilisant les mêmes primitives de synchronisation qui sont utilisées avec la programmation multithreadée (ex. mutex, sémaphore, barrière).

(4) (i) Dans le contexte du problème des généraux byzantins, expliquez en vos mots pourquoi faut-il au moins quatre généraux pour que l'on puisse fonctionner correctement, malgré la présence d'un général traître.

Solution: Si l'on a moins de quatre généraux, supposons trois généraux, et que un de ceux-ci est un traître, ceci veut dire que deux des généraux sont loyaux. Lors de l'échange des information, un général loyal peut recevoir de l'information contradictoire sur les données de l'autre général loyal, comme le général traître fournira des résultats erronés. Il serait alors impossible de déterminer le bon résultats. Avec quatre généraux, le bon résultat sera également donné par un autre général loyal, évitant ainsi toutes ambiguïtés.

(4) (j) Donnez les différences entre l'utilisation des fonctions `shm_open` et `mkfifo` pour la communication entre processus d'une application temps réels.

Solution: Avec la fonction `shm_open`, un espace mémoire partagé pourra être établi entre plusieurs processus, permettant des accès aléatoires aux données communes. Avec la fonction `mkfifo`, le partage est plutôt limité à des lectures et écritures séquentielles des données.