Systèmes embarqués temps réel (GIF-3004)
Département de génie électrique et de génie informatique
Hiver 2018

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

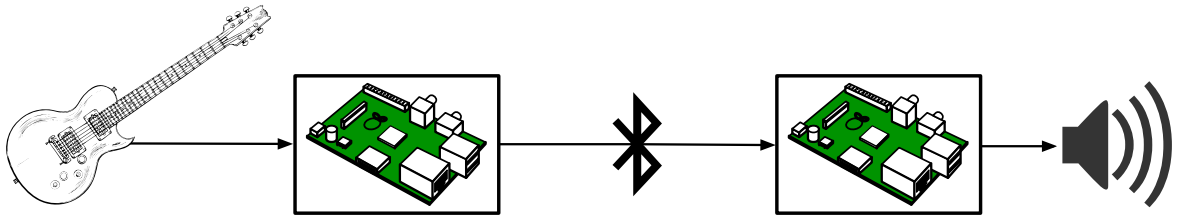
— Densité de probabilité : $p(t) = \lambda \exp(-\lambda t)$, pour $t \geq 0$
— Loi de répartition : $P(X \leq t) = 1 - \exp(-\lambda t)$, pour $t \geq 0$
— Espérance : $\mathbb{E}(t) = \frac{1}{\lambda}$
— Variance : $\mathrm{Var}(t) = \frac{1}{\lambda^2}$

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 (52 points sur 100)

Soit le contexte du laboratoire 5 du cours, avec un système d'acquisition sonore et transmission sans fil basé sur deux *Raspberry Pi* (RPi) et un lien de communication Bluetooth.



Supposons la caractérisation suivante du système :

— Temps d'acquisition du son de $T_{\text{acq}} = 30\,\text{ms}$;
— Temps de traitement et transmission sur le RPi d'acquisition de $T_{\text{send}} = 50\,\text{ms}$;
— Lien sans fil Bluetooth avec bande passante de $B = 80\,\text{ko/s}$;
— Temps de propagation pour transmettre 1 bit sur le lien sans fil de $T_{\text{delay}} = 20\,\text{ms}$;
— Temps de réception et traitement sur le RPi de sonorisation de $T_{\text{recv}} = 40\,\text{ms}$;
— Temps de rendu sonore sur les haut-parleurs de $T_{\text{play}} = 20\,\text{ms}$.

(5) (a) Supposons un lien sans fil orienté flot (datagram), où le son est transmis immédiatement sur le lien réseau aussitôt que la donnée est disponible, dans un flot continu. Déterminez la latence bout à bout du système, entre le moment où un son produit par l'instrument de musique et le moment où ce son est entendue sur les haut-parleurs.

Solution: La latence bout à bout devrait correspondre à la valeur suivante :

$$\begin{aligned} T_{\text{dgram}} &= T_{\text{acq}} + T_{\text{send}} + T_{\text{delay}} + T_{\text{recv}} + T_{\text{play}} \\ &= 30 + 50 + 20 + 40 + 20 = 160\,\text{ms}. \end{aligned}$$

(14) (b) Supposons maintenant un contexte plus réaliste d'un lien sans fil orienté connexion, où des paquets de taille fixe sont transmis sur le lien entre le RPi d'acquisition et de sonorisation. Les données acquises sont divisées en blocs de taille fixe (paquets), avec compression des données, somme de contrôle (*checksum*) pour vérification d'intégrité et d'accusés réception pour chaque paquet. Pour les besoins de l'exercice, vous pouvez présumer que les paquets utilisés pour la compression et pour la transmission Bluetooth sont exactement les mêmes (même taille et même découpage des données).
Voici trois configurations possibles pour le lien sans fil orienté connexion ($1\,\text{ko} = 1024\,\text{o}$).

Configuration	A	B	C
Quantité de données par paquet	$64\,\text{o}$	$256\,\text{o}$	$1024\,\text{o}$

Pour chaque paquet, supposez que l'on doit ajouter l'équivalent de $64\,\text{o}$ de données supplémentaires pour le protocole de communication et de compression. Faites l'analyse de ces trois configurations et déterminez laquelle est la plus intéressante pour réduire la latence bout à bout, en supposant qu'une bande passante effective de $50\,\text{ko/s}$ est nécessaire pour obtenir la qualité sonore minimale requise avec compression.

Solution: La latence bout à bout est la suivante :

$$T_{\text{packet}} = T_{\text{acq}} + T_{\text{send}} + T_{\text{delay}} + \frac{m_{\text{packet}} + m_{\text{overh}}}{B} + T_{\text{recv}} + T_{\text{play}}.$$

On voit que celle-ci sera directement minimisée par une taille minimale de paquets (minimiser m_{packet}), les autres variables étant les mêmes pour toutes les configurations. Cependant, on doit évaluer la bande passante effective de chaque configuration pour déterminer si elle remplit les exigences minimales de $50\,\text{ko/s}$. La bande passante effective se calcule selon l'équation suivante :

$$B_{\text{eff}} = B \times \frac{m_{\text{packet}}}{m_{\text{packet}} + m_{\text{overh}}}.$$

Pour la configuration A :

$$B_{\text{eff}}^{A} = 80 \times \frac{64}{64 + 64} = 40\,\text{ko/s} < 50\,\text{ko/s}.$$

Pour la configuration B :

$$B_{\text{eff}}^{B} = 80 \times \frac{256}{256 + 64} = 64\,\text{ko/s} > 50\,\text{ko/s}.$$

Pour la configuration C :

$$B_{\text{eff}}^{C} = 80 \times \frac{1024}{1024 + 64} = 75\,\text{ko/s} > 50\,\text{ko/s}.$$

La configuration B est donc celle qui minimisera la latence, tout en ayant la bande passante effective minimale requise.

(14) (c) Utilisons la configuration B de la question précédente pour la suite, soit des paquets de $256\,\text{o}$. Supposons qu'il y ait parfois des défaillances du lien sans fil de sorte que des paquets transmis peuvent être corrompus, ce qui exige de les transmettre de nouveau. Chaque fois qu'une telle défaillance survient, un délai additionnel de $80\,\text{ms}$ est nécessaire entre la réception du paquet corrompu par le RPi de sonorisation et le renvoi de ce paquet sur le réseau par le RPi d'acquisition. La probabilité de défaillance de chaque paquet est de $0{,}5\,\%$. Calculez la latence bout à bout moyenne du système, en faisant l'hypothèse simplificatrice qu'un paquet ne peut être corrompu qu'une seule fois. Comparez la latence bout à bout en cas de paquet corrompu avec la latence bout à bout moyenne et discutez brièvement du résultat dans un contexte de système temps réel de transmission du son.

Solution: Convertissons d'abord la bande passante effective de la configuration B dans les bonnes unités :

$$B = 80\,\text{ko/s} = 80\,\text{ko/s} \times \frac{1024\,\text{o/ko}}{1000\,\text{ms/s}} = 81{,}92\,\text{o/ms}.$$

Sans défaillance, la latence bout à bout est la suivante :

$$\begin{aligned} T_{\text{packet}} &= T_{\text{acq}} + T_{\text{send}} + T_{\text{delay}} + \frac{m_{\text{packet}} + m_{\text{overh}}}{B} + T_{\text{recv}} + T_{\text{play}} \\ &= 30 + 50 + 20 + \frac{256 + 64}{81{,}92} + 40 + 20 = 100 + 3{,}91 + 80 = 183{,}91\,\text{ms}. \end{aligned}$$

Avec une défaillance, la latence bout à bout est :

$$\begin{aligned} T_{\text{fail}} &= T_{\text{acq}} + T_{\text{send}} + 2 \times \left(T_{\text{delay}} + \frac{m_{\text{packet}} + m_{\text{overh}}}{B} \right) + T_{\text{resend}} + T_{\text{recv}} + T_{\text{play}} \\ &= 30 + 50 + 2 \times \left(20 + \frac{256 + 64}{81{,}92} \right) + 80 + 40 + 20 \\ &= 80 + 2 \times 23{,}91 + 140 = 267{,}82\,\text{ms}. \end{aligned}$$

La latence bout à bout moyenne est donc :

$$\begin{aligned} T_{\text{avg}} &= (1 - P_{\text{fail}}) \times T_{\text{packet}} + P_{\text{fail}} \times T_{\text{fail}} \\ &= (1 - 0{,}005) \times 183{,}91 + 0{,}005 \times 267{,}82 = 182{,}9905 + 1{,}3391 = 184{,}33\,\text{ms}. \end{aligned}$$

On voit donc que la latence moyenne est peu affectée par les paquets corrompus, mais que la latence en cas de paquet corrompu est significativement plus élevée que la latence moyenne — $267\,\text{ms}$ comparativement à $184\,\text{ms}$, soit une augmentation de la latence de $45\,\%$. Dans un contexte de temps réel, une telle variation est très importante comme il faut configurer le système afin qu'il respecte les contraintes temps réel dans le pire des cas, qui ici diffère significativement du cas moyen et peut impliquer des délais perceptibles à l'oreille humaine.

(14) (d) Supposons maintenant que l'on utilise une modélisation plus sophistiquée de la corruption de paquets sur le réseau, selon une file d'attente de type M/M/1, de sorte que plusieurs paquets peuvent être corrompus simultanément. On suppose alors un taux de corruption (d'arrivée) de $\lambda = 1$ paquet corrompu par seconde et un taux de traitement (départ) de $\mu = 5$ retransmissions par seconde. On suppose que le tampon utilisé permet de tolérer jusqu'à deux paquets corrompus simultanément. Calculez la disponibilité du système relativement à cette modélisation. Indiquez également si cette disponibilité est satisfaisante pour le fonctionnement du système.

Solution: On va calculer les probabilités qu'il y ait aucune, une ou deux corruptions en cours de traitement par la file d'attente de type M/M/1 :

$$\begin{aligned}\rho &= \frac{\lambda}{\mu} = \frac{1}{5} = 0{,}2,\\ \pi_0 &= (1-\rho) = 1 - 0{,}2 = 0{,}8,\\ \pi_1 &= (1-\rho)\rho = (1-0{,}2) \times 0{,}2 = 0{,}16,\\ \pi_2 &= (1-\rho)\rho^2 = (1-0{,}2) \times 0{,}2^2 = 0{,}032.\end{aligned}$$

La disponibilité est la somme de ces probabilités, comme le système devient non disponible avec plus de deux paquets corrompus :

$$A = \pi_0 + \pi_1 + \pi_2 = 0{,}8 + 0{,}16 + 0{,}032 = 0{,}992.$$

Cette disponibilité est plutôt faible en pratique, comme elle implique que le système sera défaillant $0{,}8\,\%$ du temps, ce qui sera certainement perceptible aux utilisateurs du système.

(5) (e) Indiquez si la modélisation par file d'attente faite à la question précédente est réaliste dans ce contexte. Justifiez votre réponse brièvement et clairement.

Solution: Cette modélisation n'est certainement pas réaliste. D'abord, le taux d'arrivée de paquets corrompus ne peut pas suivre réalistement une loi exponentielle, comme les paquets seront transmis sur le réseau à intervalle régulier alors que selon le modèle, les corruptions peuvent survenir à tout moment. Ensuite, le taux de traitement peut difficilement suivre une loi exponentielle, qui implique des temps de retransmission qui pourraient être très courts, trop courts pour être réalisés en pratique étant donné les délais du réseau.

Question 2 (48 points sur 100)

Répondez aussi brièvement et clairement que possible aux questions suivantes.

(4) (a) Dans la présentation des 17 règles de la philosophie Unix, une des règles présentées est celle de la **représentation**. Expliquez en quoi consiste cette règle.

Solution: La règle de la représentation vise à placer la connaissance dans les données afin de rendre la logique du programme stupide et robuste. Ceci s'appuie sur le principe qu'il est plus facile de raisonner sur (et de déboguer) des structures de données complexes que des procédures complexes. La complexité doit donc être déplacée autant que possible des programmes vers les structures de données.

(4) (b) Indiquez si Linux correspond à un modèle de noyau monolithique ou de micro noyau, en précisant ce que cela implique sur l'architecture du noyau.

Solution: Linux est un noyau monolithique, se présentant comme un seul gros programme s'exécutant en mode privilégié, avec l'ensemble de l'espace mémoire partagé dans le noyau. Les échanges entre les composantes du noyau se font par des appels de fonction.

(4) (c) Expliquez ce qui rend Linux similaire à un micro noyau relativement aux pilotes de périphériques.

Solution: Le chargement et déchargement dynamique de modules logiciels est possible dans Linux, permettant ainsi d'ajouter des composantes nouvelles dans le noyau, telles que le chargement de pilotes de périphériques, qui peuvent ainsi être chargés à la volée, lors de branchement ou l'activation de nouveaux périphériques. Ceci fait contraste avec les SE à noyau monolithique plus classiques, où les pilotes doivent être compilés à l'avance dans le noyau, et rapproche Linux du fonctionnement de SE à micro-noyau, où les pilotes peuvent être lancés et chargés à la volée, dans des processus distincts du processus principal du micro noyau.

(4) (d) Expliquez pourquoi il n'est pas possible de faire appel aux fonctions standards du langage C dans le noyau Linux.

Solution: Les fonctions standards du langage C exigent souvent des fonctionnalités offertes par le noyau, qui ne sont pas disponibles dans le noyau en soi. Également, ces fonctions peuvent parfois être trop grosses et inefficaces pour être utilisées dans le noyau. Cependant, le noyau Linux implémente un ensemble de fonctions utilitaires similaires aux fonctions standards C pouvant être utilisées dans le code du noyau (ex. `printk`, `kmalloc`).

(4) (e) Quel est l'intérêt d'utiliser un *spinlock* (verrou tournant) relativement à un mutex dans le noyau.

Solution: Le *spinlock* est utile pour une courte attente active sur une ressource dans des sections critiques du noyau (ex. gestion des interruptions), comparativement à l'utilisation de mutex pour des sections où l'attente peut être longue.

(4) (f) Expliquez pourquoi le traitement des interruptions par les pilotes est souvent séparé en deux parties, soit la moitié inférieure (*bottom half*) et la moitié supérieure (*top half*).

Solution: Le traitement des interruptions exige que certaines sections critiques soient traitées rapidement. Cependant, il faut faire attention de ne pas monopoliser les ressources pour certains traitements plus longs. C'est pourquoi on sépare les traitements en deux parties, soit la moitié supérieure, pour des traitements courts, immédiats et non préemptibles souvent associés aux événements matériels (ex. gérer une interruption matérielle), et la moitié inférieure, qui se fait dans un contexte courant et préemptible du noyau pour des opérations plus complexes et longues, évitant ainsi de bloquer inutilement le traitement des autres opérations du noyau plus critiques.

(4) (g) Expliquez la différence entre les dispositifs de caractères (*character devices*) et les dispositifs de blocs (*block devices*) comme classes de dispositifs dans Linux.

Solution: Les dispositifs de caractères sont utilisés pour des périphériques associés à des flots séquentiels de données, où ces données sont généralement lues ou écrites dans l'ordre, comme des claviers ou ports série. Des dispositifs de blocs permettent eux de faire des accès aléatoires à des morceaux de taille fixe (blocs), ce qui correspond à des dispositifs comme des disques durs ou mémoires flash.

(4) (h) Indiquez quel type d'infrastructure de communication devrait-on établir entre des dispositifs dans des systèmes temps réel dur, en justifiant votre réponse.

Solution: On devrait favoriser une infrastructure avec liens dédiés, par exemple de type point à point, permettant ainsi d'avoir certaines garanties sur les performances des communications pour respecter les délais. Dans des infrastructures réseau partagées avec routage de paquets, par exemple, les délais de communication peuvent varier significativement selon l'ampleur du trafic sur le réseau, ce qui est souvent difficile à contrôler aisément sans dédier des ressources au système (en obtenant un accord de qualité de services respecté par l'ensemble du réseau de communication).

(4) (i) 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) (j) Soit un programme où 20 % du temps d'exécution équivalent sur un processeur doit s'exécuter sur un nœud central, le reste du programme pouvant être exécuté en parallèle sur un grand nombre de nœuds de traitement. Expliquez pourquoi l'accélération (*speedup*) du programme ne peut pas excéder 5.

Solution: Selon la loi d'Amdahl, l'accélération peut se calculer selon l'équation suivante :

$$S_p(n) = \frac{T^*(n)}{f \times T^*(n) + \frac{(1-f)}{p} \times T^*(n)} = \frac{1}{f + \frac{1-f}{p}},$$

avec f comme la partie séquentielle du programme et p le nombre de processeurs alloués au programme. Avec un grand nombre de processeurs, l'accélération maximale serait :

$$\lim_{p\to\infty} S_p(n) = \lim_{p\to\infty} \frac{1}{f + \frac{1-f}{p}} = \frac{1}{f}.$$

Ceci veut dire simplement que l'accélération est bornée par la partie séquentielle du programme, la partie parallèle devenant négligeable avec une exécution massivement parallèle. Donc, avec $f = 0{,}2$, on obtient que

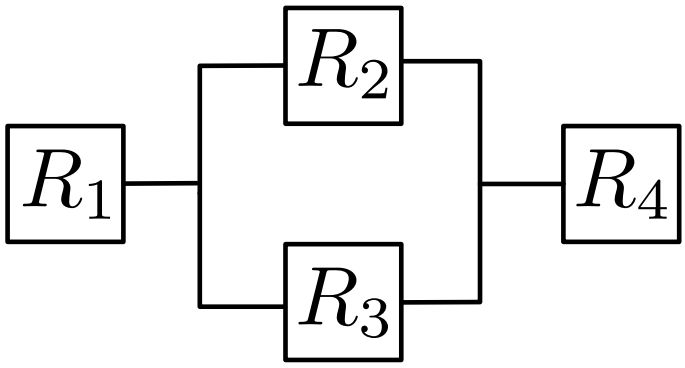
$$\lim_{p\to\infty} S_p(n) = \frac{1}{f} = \frac{1}{0{,}2} = 5.$$

Donc, 5 est l'accélération maximimale possible avec une exécution massivement parallèle du programme.

(4) (k) Supposons que l'on veut modéliser le fonctionnement d'un serveur Web relativement au nombre de requêtes reçues et traitées. Si on suppose que le serveur comporte un seul processus de traitement et un tampon pouvant conserver jusqu'à N requêtes simultanément (incluant celle présentement traitée), avec des temps d'arrivée des requêtes et des temps de traitement suivants des processus de Poisson, indiquez à quel modèle de file d'attente ceci correspond.

Solution: Ceci correspond à un modèle de file d'attente de type M/M/1/K, avec un serveur et jusqu'à $K = N$ requêtes pouvant être accommodées simultanément.

(4) (l) Donnez la fiabilité correspondant au système suivant, formé de quatre composants dont la fiabilité individuelle est donnée par R_i.



Solution: On va d'abord réduire le sous-système parallèle des composants 2 et 3 :

$$R_2 \| R_3 = 1 - \prod_i (1 - R_i) = 1 - (1 - R_2)\,(1 - R_3) = R_2 + R_3 - R_2\,R_3.$$

On a ensuite un système formé de trois éléments en série :

$$R_s = \prod_i R_i = R_1(R_2 + R_3 - R_2\,R_3)R_4.$$

Ce qui correspond à la fiabilité du système.