Denis Morency

De: no-reply@www.usherbrooke.ca
Envoyé: 14 avril 2015 09:59
À: Sciences-CentreImpression@USherbrooke.ca
Objet: COMMANDE EXAMENS
Pièces jointes: IFT729_Final_H2015.pdf

TYPE-EXAMEN	FINAL
SIGLE-COURS	IFT729
TITRE-COURS	Conception de système temps réel
PROFESSEUR	Patrice Roy
DATE-HEURE	Lundi 20 avril à 9 h
AUTORISE-PAR	Gabriel Girard et Patrice Roy
NOMBRE-PAGES	20
NOMBRE-COPIE-PROF	25
IMPRESSION-QUESTIONNAIRE	Recto-verso broché
NOMBRE-FEUILLES-BLANCHES	2
NOMBRE-PAPIER-GRAPHIQUE	
NOMBRE-CAHIERS	
CONSENTEMENT-AGES	1
REMARQUES	Patrice.Roy@usherbrooke.ca
E-MAIL	
FIRST-NAME	
LAST-NAME	
NICK-NAME	
SPAMSHIELD	true

Nom : _____ (_____/100)

Matricule : _____

Signature : _____

STR – Contrôle final

Lundi, le 20 avril 2015, de 9 h à 12 h, local D3-2034

Ceci constitue le contrôle final pour le cours ***IFT729 – Conception de systèmes temps réel*** (STR). Il est présumé, avant d'attaquer la présente, que vous avez pris une part active à l'intégration des concepts du cours dans votre projet de session et que vous avez pris sur vous de faire les exercices qui, parmi ceux proposés, vous auront aidés à parfaire votre compréhension des éléments de matière qui vous semblaient un peu plus nébuleux.

Ce contrôle est récapitulatif, ce qui signifie qu'il couvre l'ensemble de la session (ce qui ne veut pas dire qu'il en couvre chaque parcelle, l'espace et le temps n'étant pas élastiques à ce point). En conséquence, gérez votre temps intelligemment.

Les consignes pour ce contrôle sont :

- le droit aux notes de cours est accordé;
- l'envoi de notes écrites, par télépathie, via pigeons voyageurs, hiboux, ou à l'aide de tout autre mode de communication, est interdit;
- la calculatrice est bannie (en fait, vous pouvez l'utiliser mais ça ne donne honnêtement pas grand-chose), de même que le téléphone cellulaire, l'ordinateur portatif, le *Walkie Talkie* et leurs différents dérivés;
- si vous bloquez sur une question, alors passez à la suivante. Il est plus sage de revenir plus tard sur ce qui vous pose problème, s'il vous reste du temps;
- les questions de ce contrôle n'ont pas toutes le même poids. Planifiez la manière dont vous investirez temps et efforts;
- à moins d'indications contraires, vous pourrez répondre par du texte, du code, des schémas, etc., mais je ne pourrai corriger convenablement que si vos réponses sont claires. Il est donc à votre avantage de soigner le tout, et d'être aussi précis et limpide que possible;
- ceci est un examen pour gens équilibrés;
- répondez aux questions, simplement. Chaque réponse sera corrigée en fonction de l'énoncé de la question, alors assurez-vous de bien répondre à ce qui est demandé, pas à ce qui aurait peut-être pu l'être. Évidemment, justifiez vos réponses!
- pas de stress, la vie est belle. Sans blagues. J'ai beaucoup apprécié ma session avec vous;
- savourez vos vacances… ou votre carrière… ou la poursuite de vos études… ou…
- je vous souhaite une fin de session pleine d'énergie!
- répondez directement sur le document, aux endroits prévus à cet effet.

Vous trouverez une annexe à la toute fin, un aide-mémoire, juste au cas… De même, l'examen permet d'accumuler 110 points, juste au cas où vous accrocheriez sur une question, mais la note finale sera plafonnée à 100.

Que de plaisir en perspective!

Q00	Vers l'infini… et plus loin encore!	(_____/30)

Les joies des études universitaires étant désormais derrière vous, le milieu de la R&D avec accent sur le 'D' vous a ouvert ses portes. C'est bien connu, l'un des créneaux les plus importants au Québec est celui de l'aérospatiale, et c'est avec la firme B.A.R., pour *Beneficial Aerospace Research*, que vous avez choisi de vous investir.

Une fois l'accueil fait selon les usages, avec petit café et rencontre des nouveaux collègues, on vous offre un tour des installations et on vous présente ce qui vous occupera pour le proche futur : le Promeneur®, un véhicule robotisé nouveau genre destiné à naviguer de manière semi-autonome sur la planète Mars.

Vous vous permettez un gag sur la « B.A.R. Mars », mais il est clair que vos collègues l'ont déjà entendu. Vous changez de sujet.

Figure 1Le Rover(Curiosity)

À l'image du *Rover*[1] qui circule déjà sur la planète rouge, le Promeneur® sera en partie autonome, en partie dirigé depuis les installations de l'Agence spatiale canadienne, à ville St-Bruno, et en partie dirigé depuis la NASA. Son rôle ira de la recherche de vie et d'eau à une exploration de la composition minérale du sol martien et du relief de ce sol.

Pour accroître son autonomie, en comparaison avec celle de ses prédécesseurs, le Promeneur® aura un moteur plus puissant et de meilleurs mécanismes de recharge. Alors que le Rover se déplace à une vitesse moyenne de $30\,{}^{m}/_{h}$ avec des pointes de $90\,{}^{m}/_{h}$, ce qui devrait lui faire parcourir environ $19Km$ pour les deux années de sa mission principale[2], le Promeneur® sera capable de se déplacer à près d'un ${}^{Km}/_{h}$.

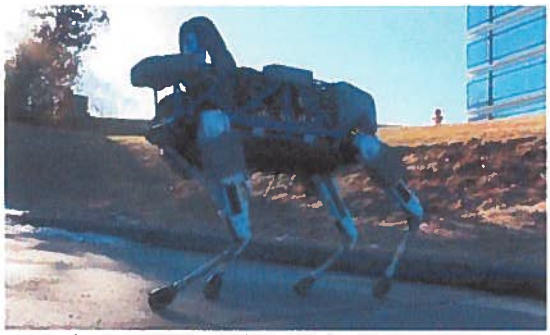

Figure 2Promeneur (inspiration de Spot, un compétiteur)

Évidemment, se déplacer à ce rythme est périlleux. La lenteur relative du *Rover* tenait en partie à la difficulté de se déplacer en terrain accidenté sans briser l'appareil ou l'embourber; en se déplaçant par petits incréments, le design de *Rover* visait à éviter les incidents et à préserver l'intégrité de l'appareil. Les avancées en intelligence artificielle et en robotique entre les époques du design du *Rover* et de celui du Promeneur® permettent d'espérer en arriver à un véhicule couvrant plus de terrain en moins de temps et nous en apprenant plus que jamais sur cette proche voisine qu'est la planète Mars.

Vous comprenez rapidement que vos connaissances en programmation, en particulier dans le créneau des STR, vous sera d'un précieux soutien dans cette nouvelle aventure.

---

[1] http://en.wikipedia.org/wiki/Curiosity_%28rover%29

[2] Inspiré de http://en.wikipedia.org/wiki/BigDog.et de https://www.youtube.com/watch?v=M8YjvHYbZ9w

Q00.0 – Sans grandes surprises, un appareil tel que le Promeneur® coûte une petite fortune. Son utilisation implique une quantité impressionnante de spécialistes, des années de préparation et de tests, et il est n'est pas raisonnable de penser aller le réparer s'il s'endommage dans le cadre de ses fonctions. Conséquemment, éviter les bris et les incidents est essentiel avec un tel objet.

Sur le plan logiciel, le Promeneur® est contrôlé par un ordinateur embarqué muni d'un Pentium 4 doté de deux cœurs et dont l'horloge est calibrée à $2\ GHz$. Ce processeur a accès à $2\ Go$ de mémoire vive, $60\ Go$ d'espace sur disque rigide et utilise le système d'exploitation QNX. L'héritage de priorités y est implémenté, ce qui vous rassure au plus haut point.

Le processeur contrôle le comportement du Promeneur®, gère les capteurs de l'appareil (incluant la cinquantaine de capteurs de mouvement) et permet de communiquer à distance. Les capteurs détectent l'altitude du Promeneur®, son accélération, la force déployée par chaque patte, etc. Cette combinaison d'informations permet de modéliser la situation du Promeneur® dans l'espace. D'autres capteurs permettent d'évaluer l'homéostasie[3] de l'appareil, la pression hydraulique, la température des composants, etc. Évidemment, des actuateurs activent les divers membres de l'appareil pour assurer son mouvement en toute sécurité. Enfin, l'ordinateur a aussi pour mandat d'enregistrer des données télémétriques pour évaluer le bon fonctionnement du Promeneur®, détecter les pannes, diagnostiquer son comportement et l'adapter au besoin.

Pour 10 points, il faudra écrire le logiciel permettant de signaler au « cerveau » du Promeneur® qu'un risque de chute est imminent. Vous savez qu'un humain prend environ $\frac{1}{4}s$ à détecter une chute, et souhaitez en arriver à ce que le Promeneur® agisse dans des délais semblables. Le temps total de réaction à considérer va du constat de la condition problématique au point où les ordres auront été donnés aux actuateurs et le plus lent d'entre eux aura commencé son action. Quels sont les paramètres dont il vous faudra tenir compte sur le plan de la programmation des processus composant les éléments clés du système tel que vous l'envisagez? Détaillez votre réponse, en tenant compte des contraintes TR logicielles et matérielles et des interactions entre les divers composants du système : _____

[3] La stabilité

Q00.1 – Les capteurs de mouvement installés dans votre prototype de Promeneur® produisent des données que vous souhaiteriez consommer pour les entreposer sur un média de masse et les étudier *a posteriori*, dans le but de raffiner les algorithmes homéostatiques.

Évidemment, il ne faut pas que la conservation de ces données empêche le système décrit en Q00.0 de faire son travail; nous ne voulons pas de bris! Chacun de ces capteurs produit 44000 nombres à virgule flottante encodés sur 32 bits chaque seconde. Sachant que les disques rigides choisis pour votre système sont en mesure d'écrire jusqu'à 32 *Mo* par seconde en blocs de 512 Ko, **pour 10 points**, décrivez votre stratégie pour réaliser cette tâche sans nuire aux caractéristiques TR du système : _____

Q00.2 – Un de vos collègues, moins expérimenté que vous avec les SETR, dit avoir lu que l'ordonnanceur de QNX est inéquitable et privilégie systématiquement les tâches de plus haute priorité parmi celles qui sont prêtes à s'exécuter (état *Ready*). Il dit craindre que les systèmes auxiliaires du Promeneur® ne parviennent jamais à s'exécuter. **Pour 10 points**, ses craintes sont-elles justifiées? Expliquez votre réponse. _____

Q01	Expertise et expérience	(_____/35)

L'un des informaticiens collaborant avec vous, Bertrand, vient à votre rencontre, perturbé. Il s'approche de vous d'un air dépité pour se confier.

Bertrand vous raconte qu'il vient de vivre un échange un peu rude avec un informaticien d'expérience. Leur échange avait trait aux exemples de code ci-dessous, à savoir lequel est préférable dans le contexte de votre système :

```
void consume_from
   (int capteur, float **buf, size_t &cap);
void free_buffer(float **buf);
void read_from_sensor(int which)
{
   float *tampon = {};
   size_t taille = {};
   consume_from(which, &tampon, taille);
   assert(taille > 0);
   // utiliser tampon + 0 à tampon + taille
   free_buffer(&tampon);
}
```

```
vector<float> consume_from(int capteur);
void read_from_sensor(int which)
{
   auto tampon = consume_from(which);
   assert(!tampon.empty());
   // utiliser tampon + 0 à tampon + taille
}
```

Dans ces exemples, la fonction `consume_from()` est appelée pour le capteur `which`. La fonction appelée est responsable d'allouer les ressources pour les données du tampon (peu importe quand ou comment) et d'y copier les données lues de ce capteur. À gauche, la variable taille contient le nombre de valeurs copiées dans le tampon, alors qu'à droite cette quantité est connue du conteneur qui est retourné.

Bertrand a proposé la version de droite, qui est courte et pour laquelle il s'est inspiré d'un extrait lu dans un magazine, alors que le programmeur d'expérience est d'avis que la version de gauche est préférable.

`Q01.0` – **Pour 10 points**, à quelles conditions donnerez-vous raison à Bertrand (à droite), et à quelles conditions donnerez-vous raison à son illustre collègue (à gauche)? Votre réponse doit tenir compte des caractéristiques d'un `STR` (exécution prévisible, résilience, brièveté d'exécution, basse latence lors d'un appel, etc.). _____

Q01.1 – Bertrand vous dit avoir lu qu'il existe des opérations de transtypage (des *casts*) de diverses natures en C++, mais il craint de les utiliser car on lui a dit que ces opérations ne sont pas appropriées pour un STR. **Pour 5 points**, cette crainte est-elle légitime? Prenez soin d'expliquer votre réponse.

Q01.2 – Selon Bertrand, les contributeurs de certains départements craignent le recours aux exceptions dans le code, tous langages confondus. **Pour 5 points**, quel est le possible problème associé aux exceptions dans un STR. **Pour 5 points**, renoncer aux exceptions a des conséquences. Identifiez une difficulté importante découlant du choix de ne pas utiliser d'exceptions pour souligner un problème à l'exécution du programme et expliquez brièvement votre réponse. _____

Q01.3 –Un processus à basse priorité du Promeneur® doit permettre de réaliser, au besoin, la mise à jour de composants logiciels, ce qui impliquera entre autres de modifier quelques fichiers de configuration et de verrouiller l'accès à quelques ressources du système. Il importe toutefois que les systèmes critiques, ceux qui assureront le maintien de l'intégrité physique du Promeneur®, ne perdent pas leur capacité d'agir dû à l'exécution de ces mises à jour. **Pour 10 points**, expliquez comment y arriver : _____

Q02	Questions noyautées	(_____/20)

Certains de vos collègues estiment que QNX, qu'ils connaissent peu, pourrait avantageusement être remplacé dans le Promeneur® par un système d'exploitation conventionnel tel que Microsoft Windows ou une déclinaison réputée de Linux.

Q02.0 – **Pour 5 points**, donnez un exemple d'une situation où un SETR permettrait à votre système de respecter ses contraintes, alors qu'un système d'exploitation conventionnel ne le permettrait pas : _____

Q02.1 – Un SETR repose typiquement sur un micronoyau minimaliste de très basse priorité. **Pour 5 points**, pourquoi en est-il ainsi? Et **pour 10 points**, considérant sa priorité, comment le noyau parvient-il à faire son travail malgré tout? _____

Q03	Questions techniques	(_____/15)

Q03.0 – **Pour 5 points**, décrivez un type de moteur de collecte d'ordures qu'il est raisonnable d'utiliser dans un programme TR. _____

Q03.1 – **Pour 5 points**, décrivez une différence clé entre un ordonnanceur de tâches périodiques et un exécutif.

Q03.2 – **Pour 5 points**, décrivez un ensemble de conditions permettant d'implémenter un mécanisme d'allocation dynamique de mémoire qui soit déterministe et s'exécute en temps constant, donc qui soit de complexité $O(1)$.

Q04	Quelques petits trucs rapides	(_____/10)

Q04.0 – **Pour 2 points**, donnez un désavantage de l'acte de déplacer des calculs du moment de l'exécution d'un processus vers celui de sa compilation : _____

Q04.1 – **Pour 2 points**, décrivez sommairement une technique par laquelle il est possible d'éviter le faux partage : _____

Q04.2 – **Pour 2 points**, quand préférera-t-on une assertion statique à une assertion dynamique? _____

Q04.3 – **Pour 2 points**, pourquoi les variables atomiques doivent-elles typiquement être d'une taille inférieure ou égale à celle du mot mémoire d'un ordinateur. _____

Q04.4 – **Pour 2 points**, pourquoi une implémentation morcelable d'un algorithme est-elle susceptible de prendre plus de temps à mener sa tâche à terme que son équivalent s'exécutant d'un seul tenant (sans morcellement)? _____

Annexe 00 – Aide-mémoire

Les unités de mesure utilisées dans les questions de cet examen vont comme suit. Supposez huit bits par *byte* (donc un *byte* équivaut à un octet), même si cela ne s'avère pas toujours en pratique.

Unité	Sens
Bit	Unité élémentaire, valant `0` ou `1`
Bps	Bits par seconde
GHz	Gigahertz. Mesure de fréquence ($n\ GHz$ signifie $n \times 1024^3$ fois par seconde)
Go	Gigaoctet. Mesure de capacité ($n\ Go$ signifie $n \times 1024^3$ octets)
Hz	Hertz. Mesure de fréquence ($n\ Hz$ signifie n fois par seconde)
Km	Kilomètres
$Km/_h$	Kilomètres
Ko	Kilooctets. Mesure de capacité ($n\ Ko$ signifie $n \times 1024$ *bytes*)
Mo	Mégaoctet. Mesure de capacité ($n\ Mo$ signifie $n \times 1024^2$ octets)
m	Mètres
$m/_s$	Mètres par seconde
Octet	Bloc de huit bits (souvent synonyme de *byte*, bien que ce ne soit pas formellement exact)
s	Secondes

Les opérations de transtypage (les *casts*) standards pour traiter une expression `expr` comme étant d'un type `T` avec C++ sont :

- les transtypages du langage C, de la forme `(T)expr`
- les appels de constructeurs, de la forme `T(expr)` et `T{expr}`
- l'opérateur `static_cast<T>(expr)`
- l'opérateur `dynamic_cast<T>(expr)`
- l'opérateur `const_cast<T>(expr)`
- l'opérateur `reinterpret_cast<T>(expr)`