UNIVERSITÉ DE SHERBROOKE

Faculté des sciences

Département d'informatique

IFT729

Conception de systèmes temps réel

Plan de cours

Hiver 2015

Enseignant : Patrice Roy
Courriel : Patrice.Roy@USherbrooke.ca
Local : D4-1010-14
Téléphone : poste 65561
Site Web pour le cours : http://h-deb.clg.qc.ca/UdeS/STR/

Note : Privilégiez (de loin!) le courriel (réponses rapides, enseignant matinal)

Horaire : Lundi, 9h30 à 12h20 salle D4-2022

Description officielle de l'activité pédagogique[1]

Objectif(s)	Connaître et identifier les problèmes inhérents au développement de systèmes temps réel; connaître et appliquer le traitement du temps au niveau des systèmes informatiques; spécifier, concevoir, programmer et vérifier des systèmes temps réel.
Contenu	Types de systèmes temps réel. Représentation du temps, contraintes de temps, horloge, synchronisation d'horloges. Formalismes utilisés dans la spécification de systèmes temps réel : machines à états, *statecharts*, réseaux de Petri, Grafcet. Approche axiomatique de spécification de contraintes temporelles. Architecture des systèmes temps réel. Acquisition et traitement de l'information en temps réel. Modèles utilisés dans la conception de systèmes temps réel : modèles basés sur les événements, modèles basés sur les graphes, modèles des tâches, modèles des processus, modèles du contrôle. Programmation d'applications.
Crédits	3
Organisation	3 heures d'exposé magistral par semaine
	6 heures de travail personnel par semaine
Préalable :	IFT 320 ou l'équivalent

[1] Tiré de http://www.usherbrooke.ca/fiches-cours/ift729

Avant d'entreprendre ce cours…

Le cours présume chez l'étudiant(e) une connaissance de la programmation procédurale et orientée objet (OO), de même qu'une solide base avec les langages C, C++ et Java.

Aussi, bien que ce cours ne soit pas strictement un cours de programmation de systèmes à haute performance, cet élément ne nous sera pas étranger.

Enfin, le cours présume chez l'étudiant(e) des aptitudes analytiques se prêtant à la spécialisation et à l'optimisation de processus. Par nature, ce cours sera régulièrement porté vers la définition de contraintes de performance et vers la mesure du respect de ces contraintes.

Il est hautement probable que ces différents *a priori* soient acquis par chacune et chacun d'entre vous, mais personne n'est parfait. Pour les travaux, envisagez fortement vous adjoindre des coéquipières et des coéquipiers ayant des atouts qui seront *complémentaires* aux vôtres. Cela activera la discussion.

Oui, mais j'ai déjà vu cela…

Il y aura inévitablement des éléments de votre formation, dans ce cours comme ailleurs, qui recouperont votre vécu académique (ou professionnel, dû à des stages ou à une expérience de travail antérieure).

De même, dû à des variations selon les parcours académiques et professionnels de chacun(e), certain(e)s parmi vous trouveront évident ce que d'autres estimeront lourd (ou cauchemardesque).

Sachant cela, mettons de l'eau dans notre vin (pas trop, quand même, pour qu'il garde son cachet) et soyons tolérant(e)s les un(e)s envers les autres.

Mise en contexte

Les STR constituent un sujet à la fois important et méconnu. Assimilés, souvent à tort, à l'idée de systèmes *très rapides*, ou à celle d'interface personne/ machine répondant *instantanément* aux demandes d'un usager, les STR sont perçus par le grand public à travers une lorgnette floue, inexacte.

J'utiliserai **TR** pour ***[en] temps réel***; STR pour ***système(s) TR***; et **SETR** pour ***système d'exploitation TR***. Cela allégera l'écriture.

Pourtant, un STR n'est pas un jouet : on attend d'un STR à ce qu'il respecte certaines contraintes de performance, exprimée en temps d'exécution de tâches précises, et ce même dans le pire cas. Un STR n'est pas tant un système rapide globalement qu'un système *en tout temps assez rapide* (ou *jamais trop lent*, ce qui revient au même).

Parfois, la contrainte TR à respecter sera simple à exprimer : une pulsation au pire chaque *n*ième quantum de temps; une écriture sur disque d'une quantité donnée d'information à chaque seconde dans le pire cas; au minimum six lectures de touches au clavier par seconde; un rafraîchissement complet d'écran au moins chaque soixantième de seconde; *etc*. Évidemment, *exprimer* et *garantir* sont deux verbes de nature bien différente.

La conception des STR constitue un sujet plus complexe qu'il n'y paraît sur les plateformes contemporaines, du fait que la majorité d'entre elles sont multiprogrammées. Il est en général impossible d'offrir des garanties assez fortes pour qu'un système puisse être qualifié de STR sur une plateforme comme *Microsoft Windows* ou Linux, par exemple, ce qui ne signifie pas que les plateformes ne permettent pas l'écriture de code performant dans l'ensemble. Ceci met en relief la distinction entre TR *souple*, qui rejoint l'idée que se fait la majorité des gens des STR (quelque chose d'instantané en apparence), et TR strict, qui est d'un autre ordre : le non-respect de contraintes TR souples entraîne un désagrément, mais le non-respect de contraintes TR strictes peut entraîner des bris matériels ou des blessures.

Le mode des STR est un monde de praticiens, même chez les théoriciens du domaine. Règle générale, un STR correspond à un besoin réel, défini de manière précise et mesurable. Un STR, surtout s'il est strict, a pour particularité que si ses contraintes ne sont pas rencontrées, le fruit de son travail n'est pas valide. *Les résultats de l'exécution d'un STR dépendent autant de la validité de ses calculs que du respect de ses contraintes de temps*. C'est d'ailleurs la définition canonique, en quelque sorte, des STR; nous explorerons aussi d'autres définitions, d'autres visions du sens à accorder au terme STR.

Enfin, les STR ont ceci de particulier qu'ils s'inscrivent systématiquement dans une démarche architecturale : une modification à un endroit, par exemple pour fins d'optimisation, peut entraîner un non-respect des contraintes ailleurs dans le même système.

Objectif général

Le cours IFT729 vise à développer chez l'étudiant(e) une **connaissance appliquée** de la conception (et du développement) d'un STR.

Conséquemment, ce cours mettra à la fois l'accent sur les connaissances permettant de comprendre ce qu'est (et ce que n'est pas) un STR, sur les techniques et concepts propres au développement d'un tel système, et sur le développement du *savoir-faire* requis pour mettre la main à la pâte ou pour encadrer une équipe de développement faisant face aux problèmes caractéristiques d'un STR.

L'approche appliquée, même pour un cours ouvert aux étudiant(e)s de 2e cycle universitaire, vous permettra d'assurer votre crédibilité face à vos pairs et à celles et ceux qui travailleront pour vous. Tel que mentionné plus haut, le monde des STR est à la fois un monde de théoriciens et de praticiens.

Nous voulons que l'étudiante et l'étudiant ayant réussi IFT729 soit capable de voir ce qui permettra (ou non) de mettre au point un STR en tout ou en partie; de débattre avec ses pairs des approches qui rapprocheront son projet et son équipe du succès; de définir les contraintes à respecter et de mesurer quantitativement l'atteinte des résultats souhaités.

Ceci sera peut-être contre-nature pour certaines et certains d'entre vous, mais il arrivera que le respect des contraintes de TR d'un module implique des choix politiques qui réduiront la performance d'autres parties du même système.

Un STR *peut* être moins performant globalement qu'un système qui ne prétend pas être un STR puisque le STR doit être drastique dans son respect de contraintes locales alors qu'un système qui ne serait pas un STR peut se préoccuper de performance au sens plus large.

Le simple fait de devoir comprendre et défendre cette réalité face à vos pairs et à vos employeurs justifie l'approche à la fois théorique et pratique du cours.

Optiques retenues

La tradition des STR est associée de près à celle des systèmes à haute performance, même s'il n'existe pas d'adéquation stricte entre les deux mondes. Certaines tâches TR (souples) peuvent être réalisées sur des plateformes grand public (*Microsoft Windows*, une distribution conventionnelle de *Linux*, peu importe), mais d'autres tâches TR (strictes) requièrent un véritable SETR. Nous partagerons notre temps entre ces deux types de plateformes. Le SETR que nous utiliserons, *QNX Neutrino*, est un système essentiellement conforme[2] à la norme POSIX ce qui suggère qu'une forte familiarité avec les langages C et C++ soit un atout.

Il est presque impossible de discuter de systèmes contemporains complexes sans adresser la question de la multiprogrammation. Nous devrons donc lui accorder une place non négligeable.

Le monde de l'informatique contemporaine accorde une place importante aux systèmes pris en charge, qui sont intrinsèquement indéterministes sur le plan performance (ces systèmes utilisent des moteurs de collecte automatique d'ordures). Nous prendrons soin d'examiner comment une plateforme comme Java peut prétendre se décliner sous une forme TR, et à quel prix.

[2] Ceci est un abus de langage; la norme POSIX est en fait une « soupe » de normes respectées à divers niveaux par divers systèmes d'exploitation, mais il demeure qu'un système se disant POSIX expose généralement une API reconnaissable à celles et ceux qui sont familières et familiers avec les systèmes UNIX et Linux.

Objectifs spécifiques

Au terme du cours, l'étudiant(e) sera capable :

{1} de distinguer un STR d'un système qui ne l'est pas
{2} d'expliquer les caractéristiques inhérentes à un STR;
{3} de définir la spécification de contraintes TR;
{4} de mesurer le respect de contraintes TR;
{5} de développer un module d'un STR;
{6} d'utiliser des primitives d'un système d'exploitation lors de la programmation de STR.

Plusieurs objectifs plus contextuels seront rencontrés en chemin : penser un programme dans un contexte de ressources limitées (les systèmes embarqués, par exemple, qui sont sujets à être soumis à des contraintes TR); modifier un moteur de collecte d'ordures (typique des STR sur plateforme prise en charge); appliquer des techniques d'optimisation locales et systémiques; développer des systèmes dans une optique de résilience et de tolérance aux pannes; *etc*.

Organisation du cours

Idéalement étalé sur environ 15 semaines, mais condensé sur 13 semaines dû aux affres du calendrier, le cours sera exigeant du point de vue de la charge de travail (certaines idées s'assimilent mieux par la pratique que par osmose). Des livrables seront exigés en chemin. Le cours comprenant un volet technique important, l'étudiant(e) devra mettre la main à la pâte et y aller d'un effort pratique.

Nous aborderons les STR sous plusieurs angles (concepts, façons de faire, comparatifs, technologiques), ce qui pourra donner une impression de désorganisation par moment (ne vous en faites pas, cependant) mais permettra un tour d'horizon plus complet.

Certains aspects plus généraux du développement informatique seront abordés au passage : par exemple, nous chercherons non seulement à rédiger des modules réalisant le travail demandé, mais aussi du code à la fois performant et portable (la portabilité du code n'est pas tant un critère spécifique aux STR qu'un atout pour votre carrière).

Certains aménagements seront possibles en cours de session, par exemple pour explorer du côté des systèmes embarqués, en fonction du matériel disponible. Le semainier qui suit est donc illustratif plutôt que contraignant.

Planification hebdomadaire

La planification hebdomadaire ci-dessous se veut un aperçu plus précis du cours tel que planifié. Cette planification se veut souple : selon le rythme que nous parviendrons à maintenir, des débordements seront possibles d'un cours à l'autre. Il est possible aussi que les questions en classe nous amènent à changer nos plans à l'occasion.

La colonne ***Séance*** indique le numéro de la séance. Ces numéros vont de `S00` à `S14` inclusivement, pour un total (dans un monde idéal) de `15` séances, ce qui représente une approximation du déroulement de la session. Un résumé visuel du contenu prévu pour la séance apparaît aussi à cet endroit, incluant une mention indiquant l'intention d'aller ou non au laboratoire en cours de séance.

La colonne ***Contenu*** décrit brièvement le contenu prévu pour cette séance.

La colonne ***Objectifs*** indique auxquels des objectifs spécifiques du cours, présentés plus haut, la séance sera *principalement* arrimée. La notation suit la convention selon laquelle le trait d'union indique un intervalle et le point-virgule indique une énumération. Ainsi :

- l'écriture `1-3` signifie de `1` à `3` inclusivement; et
- l'écriture `4-7;9` signifie de `4` à `7` inclusivement de même que `9`.

Plusieurs séances couvriront, à leur façon, l'ensemble des objectifs généraux du cours. Ce n'est pas un accident.

Consultez le document décrivant les consignes quant au projet de ce cours, qui est joint au plan de cours, pour bien organiser votre temps.

Séance	Contenu	Objectifs
S00 – Première approche Présentation du cours Présentation du professeur Exploration des divers sens de : • Temps • Temps réel Historique, problèmes et définitions Clarification de ce qu'est (ou non) un STR STR et déterminisme STR et sécurité Définition de l'activité pédagogique	Où nous définirons les termes et concepts qui nous occuperont le reste de la session, en particulier celui du temps. **Date probable** : 5 janvier	1-2
S01 et S02 – Complexité et mesure Brève exploration : • des conteneurs standards; • des itérateurs • des algorithmes standards; • des garanties de complexité. Complexité vs temps Complexité en taille ou en espace Conteneurs maison Itérateurs maison Adaptateurs Outils de mesure portables Outils de mesure non portables Approches RAII Risques et bénéfices Quoi mesurer	Nous mettrons en place quelques brèves bases de programmation efficace à partir d'outils standards et nous discuterons de stratégies de mesure du temps écoulé. *Ces séances seront fortement axées sur la programmation en C++*. **Date probable** : 12 janvier	2-5
S03 – Contraintes Contraintes Contraintes strictes ou souples Contraintes locales ou systémiques Conséquences d'un non respect Respect préventif (externe) Respect disciplinaire (interne) Scrutation Interruptions forcées Respect et matériel Respect local vs contraintes d'ensemble	Qui dit STR dit respect de contraintes de performance. Nous prendrons donc soin d'examiner la question des contraintes. **Date probable** : 19 janvier	1-4
S04 et S05 – Multiprogrammation Multiprogrammation Threads vs processus (bref) Threads *Win32* et threads POSIX Threads C++ 11 Synchronisation et performance Mémoire partagée Algorithmes sans verrous Vision générale ou disciplinaire Systèmes multi cœurs Déterminisme et multiprogrammation	Les systèmes multiprogrammés peuvent être moins déterministes que leurs contreparties monoprogrammées. On ne pourrait discuter de STR sans aborder cette question. **Date probable** : 26 janvier	1-6

Séance	Contenu	Objectifs
S06 et S07 – Un SETR (QNX *Neutrino*) Environnement de développement Outils de synchronisation Communication entre *threads* Processus et états L'ordonnanceur et commutation Communication entre processus Pointeurs de fonctions Gestionnaire d'interruption Variantes : • autres SETR; • exécutifs TR; • solutions maison.	Nous examinerons plus en détail un SETR particulier : métaphores de programmation et de multiprogrammation, mécanismes de synchronisation et de communication, outils de mesure, approches synchrones et asynchrones, *etc*. **Date probable** : 2 et 9 février	1-6

J'aimerais, si vous l'acceptez, proposer que nous ayons une **séance normale en classe le 3 février**; incluant le minitest habituel (voir ***Évaluation des apprentissages***) puisque la formule que j'applique vous avantage si le nombre de minitests est plus élevé. Je vous indique ceci car le calendrier facultaire indique que nous ne devons pas faire d'évaluations cette semaine-là.

S08 – Entrées/ sorties Contraintes TR et E/S Traitement de signal TR Approches non bloquantes Approches multiprogrammées Impact systémique des E/S E/S très rapides Vitesse locale ou équilibre systémique Équilibre et espace	Les entrées/ sorties, comme la multiprogrammation et le matériel, compliquent la question du respect des contraintes TR. Nous nous pencherons donc sur cette question. **Date probable** : 16 février	1-6

J'aimerais, si vous l'acceptez, proposer que nous ayons une **séance en classe le 23 février**; semaine des intras. Il n'y a pas d'intra en IFT729, et utiliser la séance de cette semaine nous permettrait de tenir un rythme plus humain pour la session dans son ensemble. En échange, je vous promets d'être gentil pendant cette séance ☺

Il n'y aura pas de séance le 2 mars (semaine de relâche).

S09 – Idiomes et schémas de conception Relation et implémentation d'idiomes et de schémas de conception aux STR : • singleton; • observateur; • pimpl; • *proxy*; • ordonnanceur; *etc*.	Les schémas de conception et les idiomes de programmation sont des outils tout à fait désirables et souhaitables. Cependant, la présence de contraintes TR influencera la gamme de choix disponibles et les conséquences de ces choix. **Date probable** : 9 mars	1;5
S10 – Gestion de la mémoire Allouer ou construire Déclinaisons de new et de delete Surcharge de new et delete Allocation positionnelle Allocation assistée Arénas Allocateurs	Plusieurs STR interdisent le recours à l'allocation dynamique de mémoire, mais il est possible de contrôler finement cette mécanique. Nous verrons comment. **Date probable** : 16 mars	3-4

Séance	Contenu	Objectifs
S11 – Exécutifs Contexte (disciplinaire) Avantages et risques Concevoir un exécutif Droits et obligations Démonstration détaillée	Nous examinerons une approche pour réaliser un STR malgré un ensemble important de contraintes de complexité systémique. **Date probable** : 23 mars	1;3-4
S12 et S13 – Systèmes pris en charge Déterminisme Performance Java TR Collecte automatique d'ordures État de la recherche	Les plateformes et les langages pris en charge ont la cote aujourd'hui mais se prêtent mal, *a priori*, au développement d'un STR. Beaucoup de recherche se fait pour faciliter le pont entre les deux. **Dates probables** : 30 mars et 13 avril	1-6
S14 – Contrôle final Chic examen plein d'amour	Examen final **Période du 14 au 24 avril** (à déterminer par la Faculté des sciences)	1-6

Approche pédagogique préconisée

Pour favoriser l'intégration des nombreux concepts au menu, nous suivrons essentiellement le modèle suivant :

- exposés magistraux en classe, où les étudiant(e)s sont *fortement* encouragés à contribuer par leurs questions et commentaires;
- travaux pratiques et exercices à teneur formative, qui permettront aux étudiant(e)s de mesurer concrètement leur compréhension de la matière explorée, et qui pourront être corrigés par le professeur ou autocorrigés à l'aide d'une grille de vérification, selon le cas;
- travaux pratiques à teneur sommative, évalués par le professeur en fonction des mêmes critères que ceux appliqués dans le cadre des activités formatives. Ces travaux seront soumis à un échéancier de livrables (voir le document **Consignes quant au projet**, ci-joint);
- des questions de réflexion (et parfois à saveur technique) sur une base quasi hebdomadaire; et
- un contrôle théorique récapitulatif, en toute fin de parcours, vérifiant formellement l'atteinte des objectifs.

Les questions et contrôles présumeront que chaque membre d'une équipe a contribué activement à la réalisation de chacun des travaux pratiques, et a bien compris les implications philosophiques et techniques de ces travaux.

Évaluation des apprentissages

Les évaluations sommatives seront réparties et pondérées comme suit.

Minitests (40%)

De petits tests, souvent d'une seule question, auront lieu sur une base relativement régulière, soit une par semaine, et ce la plupart des semaines. Seuls les huit mieux réussis pour chaque étudiant(e) seront comptabilisés.

Chaque question portera sur le thème de la semaine précédente, parfois des deux semaines précédentes. Les questions seront surtout orientées sur la réflexion, mais la technique à proprement dit s'y glissera à l'occasion.

Le poids de chacun de ces petits tests sera 5% de la note finale. En général, le temps alloué pour y répondre sera d'environ 15 minutes, au début du cours.

Projet (30%)

Un projet, dont le format peut vous sembler quelque peu inhabituel, sera composé de travaux pratiques (voir document joint pour l'échéancier, les outils, la constitution des équipes de travail, la philosophie selon laquelle les notes seront attribuées, etc.) et devra être réalisé en cours de session.

Dû à la nature du cours, les efforts à consentir pour réussir cette activité seront principalement axés vers la conception, la réalisation concrète et la validation du respect des contraintes TR dans un système.

Les modalités d'évaluation et les dates de livraison des livrables sont indiquées dans le document décrivant les consignes quant au projet. Notez que le professeur peut accorder un ajustement de ± 20% en fonction de la qualité du travail.

Examen (30%)

Un examen final récapitulatif valant 30% de la note finale aura lieu lors de la dernière séance de la session.

Un document dont le texte et la structure se rapporte à des textes intégraux tirés d'un livre, d'une publication scientifique ou même d'un site Internet, doit être référencé adéquatement. Lors de la correction de tout travail individuel ou de groupe une attention spéciale sera portée au plagiat, défini dans le Règlement des études comme « le fait, dans une activité pédagogique évaluée, de faire passer indûment pour siens des passages ou des idées tirés de l'œuvre d'autrui. ». Le cas échéant, le plagiat est un délit qui contrevient à l'article 8.1.2 du Règlement des études[3] : « tout acte ou manœuvre visant à tromper quant au rendement scolaire ou quant à la réussite d'une exigence relative à une activité pédagogique. » À titre de sanction disciplinaire, les mesures suivantes peuvent être imposées :
a) l'obligation de reprendre un travail, un examen ou une activité pédagogique et
b) l'attribution de la note E ou de la note 0 pour un travail, un examen ou une activité évaluée. Tout travail suspecté de plagiat sera référé au Secrétaire de la Faculté des sciences.

[3] http://www.usherbrooke.ca/programmes/etude

Médiagraphie

Des notes de cours seront mises à votre disposition, et devraient être votre référence principale pour la session qui s'annonce.

Aucun manuel n'est obligatoire. La littérature sur les STR est surtout composée d'articles et de guides accompagnant certains SETR; nous l'utiliserons sous sa forme en ligne, ce qui sera plus pratique pour nous à bien des égards que ne le seraient des imprimés.

Le site Web du cours devrait être, avec les notes de cours en tant que telles, votre référence principale :

> http://h-deb.clg.qc.ca/UdeS/STR/

Un ensemble toujours croissant de liens portant sur les STR peut être consulté ici :

> http://h-deb.clg.qc.ca/Liens/Temps-Reel--Liens.html

Sujets connexes :

> http://h-deb.clg.qc.ca/Liens/Multiprogrammation--Liens.html
> http://h-deb.clg.qc.ca/Liens/Optimisation--Liens.html