L71 - Programmation temps réel

La programmation embarquée et temps réel en C, C++ et Java(TM).

Java est une marque déposée de Sun Microsystems

Objectifs

- Découvrir les concepts de base du multi-tâches et du temps réel
- Apprendre à lire des diagrammes de conception temps-réel UML-RT
- Comprendre le fonctionnement d'une chaîne de compilation
- Analyser les options d'optimisation de code du compilateur
- Développer des applications pour accéder aux ports d'E/S et gérer les interruptions
- Interfacer C et assembleur pour mettre en œuvre des instructions spécifiques du processeur cible
- Maîtriser les difficultés de la programmation concurrente
- Connaitre les standards applicables
- Découvrir les contraintes temps réel (déterminisme, interruptions, préemption...)
- Comprendre les implications des architectures des processeurs en contexte temps-réel (cache, pipeline,...)
- Mettre au point des applications multi-tâches
- Utiliser le langages C pour implémenter un noyau temps-réel minimal

A la demande des développeurs auxquels elle est particulièrement destinée cette formation met principalement l'accent sur de nombreux exercices pratiques allant jusqu'à la réalisation effective d'un système temps-réel embarqué.
Les personnes désireuses d'un cours couvrant de façon plus large les problèmes d'utilisation d'UML/RT dans le contexte projet peuvent regarder également notre cours référence cours L70 - Les projets temps réel

Matériel

- Un PC et une carte ColdFire par binôme
- Chaîne de compilation croisée et sonde d'émulation BDM
- Machine virtuelle Java
- Manipulations et exercices en environnements natif et croisé
- Un support de cours ainsi que la disquette contenant les exemples

Pré-requis

- Connaissance de la programmation en C, C++ ou Java (niveau cours L2, L3 ou L4)
- Connaissance d'un microprocesseur souhaitée
- Connaissance de la programmation embarquée utile

Environnement du cours

- Cours théorique
  - Support de cours au format PDF (en anglais) et une version imprimée lors des sessions en présentiel
  - Cours dispensé via le système de visioconférence Teams (si à distance)
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
  - Exemples de code, exercices et solutions

    - Pour les formations à distance:
      - Un PC Linux en ligne par stagiaire pour les activités pratiques, avec tous les logiciels nécessaires préinstallés.
      - Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique
      - Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante.
    - Pour les formations en présentiel::
      - Un PC (Linux ou Windows) pour les activités pratiques avec, si approprié, une carte cible embarquée.
      - Un PC par binôme de stagiaires s'il y a plus de 6 stagiaires.
    - Pour les formations sur site:
      - Un manuel d'installation est fourni pour permettre de préinstaller les logiciels nécessaires.
      - Le formateur vient avec les cartes cible nécessaires (et les remporte à la fin de la formation).
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque session (demi-journée en présentiel) une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Modalités d'évaluation

- Les prérequis indiqués ci-dessus sont évalués avant la formation par l'encadrement technique du stagiaire dans son entreprise, ou par le stagiaire lui-même dans le cas exceptionnel d'un stagiaire individuel.
- Les progrès des stagiaires sont évalués de deux façons différentes, suivant le cours:
    - Pour les cours se prêtant à des exercices pratiques, les résultats des exercices sont vérifiés par le formateur, qui aide si nécessaire les stagiaires à les réaliser en apportant des précisions supplémentaires.
    - Des quizz sont proposés en fin des sections ne comportant pas d'exercices pratiques pour vérifier que les stagiaires ont assimilé les points présentés
- En fin de formation, chaque stagiaire reçoit une attestation et un certificat attestant qu'il a suivi le cours avec succès.
    - En cas de problème dû à un manque de prérequis de la part du stagiaire, constaté lors de la formation, une formation différente ou complémentaire lui est proposée, en général pour conforter ses prérequis, en accord avec son responsable en entreprise le cas échéant.

Plan

Introduction au temps réel

- concepts temps réel de base
- contraintes particulières du temps réel
- programmation structurée et objet
- apports des techniques objets

L'approche temps réel avec UML

- genèse d'UML
- modèles UML standards
- cycle de développement Objet
- contraintes liées à l'interprétation des diagrammes
- comment lire les diagrammes UML

Analyse des éléments constitutifs d'une chaîne de compilation

- Explication des étapes du processus de génération de code en natif et en croisé
- Rôle du compilateur, de l'assembleur et du linker
- Paramétrage en fonction d'un mapping mémoire
- Structure d'un programme source C, distinction des parties essentielles : préprocesseur, déclaration des variables globales, fonctions
- Découpage d'une application en fichiers distincts
- Le préprocesseur

- Les instructions define et include
- Ecriture de macros
- Précautions à prendre dans les headers pour éviter les redéclarations de variables
- Notion de projet, réalisation de librairies

Exercise : *mise en œuvre de la chaîne de compilation et fabrication d'une librairie*

Particularités de la programmation dans le contexte embarqué

- Analyse d'une instruction de transfert assembleur pour comprendre l'accès à la mémoire
- Mise en évidence de la distinction adresse / contenu
- Calculs d'adresses, opérations mêlant pointeurs et adresses absolues
- Les tableaux de pointeurs

Exercise : *allocation d'un pointeur sur un port d'I/O*

- Accès aux champs d'une structure
- Déclaration de variables et de pointeurs sur type structuré
- Explication du padding imposé par les règles d'alignement : options pack du compilateur
- Les formats big et little endian
- Les structures à champ de bits : modélisation des périphériques
- Les unions : une même zone mémoire peut être envisagée de différentes manières

Exercise : *modélisation de l'UART du ColdFire au moyen d'une union et d'une structure à champs de bits et communication avec un terminal*

- Utilité des tableaux de pointeurs sur des fonctions

Exercise : *lancement d'une fonction à partir d'un tableau de pointeurs*

- Distinction entre adressage absolu et adressage relatif pour la relogeabilité du code et des données
- Réentrance des handlers d'exception
- Cstart : initialisation du pointeur de pile et mise à 0 des variables non initialisées
- Inconvénients des fonctions Setjmp et Longjmp
- Gestion de la mémoire
  - Algorithmes
  - Gestion des fuites mémoire
- Mise en EPROM d'une application

Exercise : *mise en évidence et détection de fuites mémoire*

Principe de fonctionnement d'un système Temps réel et embarqué

- Notion de tâche
- Cadencement des tâches selon leur priorité, préemption
- Sauvegarde de contexte
- Nécessité d'un tick temps réel pour déclencher les commutations de tâches

Les traitements concurrents

- Recensement des champs d'un descripteur de tâche
- Réalisation d'une structure chaînée des tâches en attente d'exécution
- Insertion d'un nouveau descripteur lors du chargement d'une nouvelle tâche
- Exemple de règles d'ordonnancement: priorité évoluant en fonction du temps
- Réorganisation de la file lors de l'invocation de l'ordonnanceur: préemption

Les interruptions

- Nécessité des interruptions dans un système embarqué
- Distinction entre déclenchement sur front et sur niveau
- Acquittement logiciel
- Ecriture d'un gestionnaire d'interruption : distinction des 3 étapes prologue / corps / épilogue
- Table de vecteurs
- Ecriture des fonctions d'installation et de lecture de vecteur

Exercise : *lancement d'une action suite à une interruption*

Mise au point

- Communication avec la cible
- Les différents niveaux de mise au point : C, assembleur

- Les fenêtres du debugger : source, mémoire, pile et variables
- Positionnement de points d'arrêt
- Analyse de la pile et extraction des stacks frames correspondant aux fonctions imbriquées
- Les procédés de mise au point sur les processeurs récents munis de cache : synchronisation avec un analyseur logique

La programmation dans le contexte multi-tâches

- Structures de données:
  - Listes simplement chaînées
  - Listes doublement chaînées
  - Listes circulaires
  - Files d'attentes
  - Piles

Exercise : *réalisation de listes chaînées utilisables en contexte multi-tâches*

- Noyau temps-réel minimal
  - descripteurs de tâches
  - scheduling

Exercise : *réalisation d'un scheduler simple, "fair scheduling"*

- Synchronisation entre tâches
  - Modes de synchronisation
  - Primitives de synchronisation

Exercise : *gestion d'un sémaphore, "fixed scheduling"*

- Gestion des accès concurrents
  - Variable simple
  - Structure de données
  - Entre tâches
  - Entre tâches et routines d'interruption

Exercise : *synchronisation et communication entre tâches*

- Gestion de la mémoire
  - Algorithmes
  - Gestion des fuites mémoire

Exercise : *mise en évidence et détection de fuites mémoire*

Renseignements pratiques

Durée : 4 jours
Prix : 1850 € HT