L70 - Les projets temps réel

Conduire un projet temps réel embarqué en C, C++ ou Java(TM).

Java est une marque déposée de Sun Microsystems

Objectifs

- Découvrir les concepts de base du multi-tâches et du temps réel
- Découvrir la méthode de conception temps-réel UML-RT
- Comprendre le fonctionnement d'une chaîne de compilation
- 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,...)

Cette formation est particulièrement adaptée aux personnes ayant à conduire des projets d'informatique embarquée et temps-réel, ou devant participer à toutes les étapes de ces projets, de l'analyse et la concepton à la réalisation.
Les personnes désireuses d'un cours plus orienté vers le développement peuvent regarder également notre cours référence cours L71 - Programmation 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.

Plan du cours

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
- interprétation des diagrammes
- définition de nouveaux diagrammes
- cycle de développement avec RT UML

Le langage de modélisation UML

- modélisation statique
    - cas d'utilisation
    - modèles de classes
- modélisation dynamique
    - diagrammes de Séquence
    - diagrammes de Collaboration
    - diagrammes Etats Transitions

Les extensions pour le temps réel

- environnement / diagramme de contexte système
- contraintes / diagramme de contraintes
- comportement / diagramme d'états
- timings / diagramme de séquence étendu
- parallélisme / diagramme architecture logicielle
- architecture / diagramme architecture matérielle

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
- 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

Particularités de la programmation dans le contexte embarqué

- Les tableaux de pointeurs
- Accès aux champs d'une structure
- Déclaration de variables et de pointeurs sur type structuré
- 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
- Utilité des tableaux de pointeurs sur des fonctions

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 standards du temps réel

- Les standards d'environnements temps réel
  - POSIX
- Les langages à sémantique temps-réel
  - Java
  - Ada

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

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

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

- 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