Notes de Cours INF 8725

Olivier Sirois

2018-10-10

Contents

Chapter 1

Chapitre 1 - Introduction

Quelques définitions:

1.1 Systeme temps réelle

Contraintes de temps très importantes, elle fait normalement partie des spécifications et doivent absolument être respecté. Normalement on parle d'un environnnement qui agit sur des capteurs (senseurs) avec le système qui agit de facon correcte à ces stimuli dans un temps donné.

Les systèmes temps réelle sont normalment composé de deux sous-systèmes:

- contrôleur (PC ou microControleur)
- contrôlé (environnement physique)

La relation entre les deux sous-système est décrite par trois opérations

- échantillonnage
- calcule
- réponse

On différencie aussi entre les contraintes dure et les contraintes douces. Normalement, il est critique de devoir respecté les contriantes dure tandis que pour les contraintes c'est un peu plus flexible.

def des slides Un systeme qui doit repondre a un stimuli provaenant d'un environnement dans un temps donnée.

1.2 Systeme embarquée

Un système embarqué est un système autonome, souvent temps réelle, servant à résoudre des fonctions et des tâches spéficique et disposant de resources limités. c'est limité parce que normalement on essaie de réduire les couts parce qu'on veut en faire une production industriel.

La complexité peut varier, c'Est pas la même chose entre un lave-vaisselle et un contrôleur de missile. Normalement, on utilise du matériel pour la performance et la consommation tandis qu'on utilise du logiciel pour la flexibilité.

1.2.1 contraintes

Normalement, les métriques de conception d'un système embarqué sont :

- Taille
- fiabilité
- Consommation et dissipation de puissance
- cout de production
- temps de commercialisation (time to market)

On peut aussi avoir d'autre contrainte, genre:

- Tolérance aux pannes
- Résistance aux chocs et temp.
- BIST - built in self test. Pouvoir se diagnostiquer automatiquement
- Flexibilité et mise-a jour

Normalement, es systèmes embarqués sont utilisé dans un environnement réactif ou dans des situation très demandant.

1.3 Class d'application

- **Systèmes dominé par le contrôle - contrôle sophistiqué** - Requiert des contrainte de temps, de temps dure(critical failure) et mêmes plusieurs tâche simultané (multitache, concurrence, changement de contexte rapide requis ¿1us).

  Normalement, il y a peu de données associsé à chaque machine à états. On peut normalement associer la mémoire sur puces à ces machine pour

accélérer le changement de contexte.

Un RTOS préemptif est généralement requis, c-a-d. un OS qui a un ordonnancement de tâche sophistiqué qui tache les plus prioritaire en premier. Plusieur algorithme d'ordonnancement existe pour ces OS

INSÉRÉ IMAGE DE SYSTEME DOMINÉS PAR CONTRÔLE

- **Systèmes dominés par les données - contrôle qui traite une quantité importante de données.** - À certaine caractéristique:
  - Beaucoup de MIPS ou de MFLOPS
  - bande passante élevé
  - instruction spécialisé pour DSP
  - Support limité pour les interruptions et les changements de contexte
  - beaucoup de données pour un même contexte
  - très peu de changement de contexte son nécessaire car un seul flot de données mais à grand débit

Chapter 2

Noyau RTOS

2.1 Système avant/arrière-plan vs système multitache

2.1.1 Arrière plan

s'applique normalement à des système peu complexe ou de petites tailles. C'est généralement une boucle infini qui appel à tour de rôle différent modules. La gestion des événements est généralement asynchrones (ISR). On peut avoir des interruptions de minuteur ou par I/O.

INSÉRÉ IMAGE DE SYSTÈME AVANT/ARRIÈRE-PLAN

2.1.2 Systeme multi-tache

S'applique généralement a des systeme plus complexes. Elle favorise la réutilisation du code de par sa modularité. On donne accès à des outils évolués (Flgas, mutex, sémaphore, MB, Q). sa permet d'augmenter l'abstraction lors de la conception et exploite le concept de tâche et de programmation concurrente. (linux kernel).

INSÉRÉ IMAGE DE SYSTEME MULTI-TACHE

tâche Un simple programme évoluant comme si il avait le CPU pour lui-même. Typiquement une boucle infini dans un des états suivant:...

Programmation concurrente

Normalement utilité our exprimer le potentiel parallélisme et pour gérer les problème de synchronisation et de communication dans les systèmes multi-

tâches.

INSÉRÉ IMAGE MULTI-TACHE

2.2 Ressources partagées

La programmation concurrente introduis nécessairement le problème de partage des ressources. Une ressource est entité exploité par une tâche. Elle peut etre corrompu si elle est simultanément utilisé, ce qui va inévitablement crasher..

Pour partager les ressources, on chercher nécessairement a avoir une fonction réentrante. une fonction qui peut etre appelé par plus d'une tâche sans crainte de corrompre les données (threadsafe).

Pour s'assurer qu'une fonction soit réentrante, on s'assure généralement que les variables sont le plus souvent locals que possible.

On exploite les mutex lors de partage de variables.

Désactiver les interruptions ou le scheduleur.. (pas recommander).

2.2.1 Sections critiques

Une section est dite critique si elle doit etre exécuter sans être interrompus. Ce qui pourrait causer un deadlock.. ou un traitement de données incorrect.