HEIG VD | REDS Institut Reconfigurable and Embedded Digital Systems

Yorick Brunet, Anthony I. Jaccard

Laboratoire de Programmation Temps Réel

Semestre printemps 2024 - 2025

Laboratoire 7 : Gestion de surcharge

Temps à disposition : 4 périodes

Le laboratoire se fait par groupe de 2 personnes.

Objectifs

Le but de ce laboratoire est d'ajouter un système de gestion de surcharge à la tâche vidéo du laboratoire précédent. Les spécifications pour la gestion de surcharge sont les suivant :

- Votre système devra être capable de détecter la surcharge, c'est-à-dire, détecter que la tâche vidéo a pris plus de temps que prévu et qu'une échéance a été manquée.
- En fonction de l'état du système, vous devez mettre en oeuvre différentes stratégies de dégradation (au moins 2) du traitement vidéo permettant de retrouver un état stable
- La détection se fera à l'aide des overruns de timer EVL et d'un système de watchdog que vous implémenterez

Rappel : Vous devez utiliser taskset afin de contraindre l'exécution des tâches sur le CPU0

Etape 1 : Observation de la surcharge

Pour ce laboratoire, nous vous fournissons un programme fonctionnel comprenant 2 thread RT :

- video_task : La tâche vidéo telle que vue dans les laboratoires précédent effectuant une convolution sur la vidéo
- load_task : Une tâche créant une surcharge artificielle du système en fonction des switchs sur la carte

Les switchs permettent de générer de la surcharge CPU grâce à une tâche de haute priorité (load_task). Celle-ci s'exécute avec une période de 100ms (10Hz) et sature le CPU en fonction de la valeur des switchs afin de simuler une autre tâche lourde

Switchs	Saturation/Temps d'exécution(ms)
0	0
1-100	1-100
101+	100

Essayez votre code pour voir ce qui se passe en cas de surcharge. ⚠ Vous ne devez pas modifier la priorité de ces tâches. Modifiez ensuite le code afin de pouvoir détecter lorsqu'une surcharge se produit en utilisant les overruns de timer et documentez la valeur minimale causant une surcharge.

Etape 2 : Détection de la surcharge avec watchdog

Afin de détecter la surcharge et même pouvoir anticiper le temps à disposition pour le traitement,

nous vous demandons d'implémenter un watchdog et son canari. Ces noms correspondent à 2 tâches périodiques supplémentaires à créer au lancement du programme. Le canari se contente d'incrémenter un compteur à une certaine fréquence que le watchdog surveille afin de vérifier que le canari est toujours "vivant" et prévenir le système si ce n'est pas le cas.

Vous implémenterez ces tâches et justifierez leur priorité et fréquence d'exécution dans le rapport. Lorsque le watchdog détecte une surcharge, il devra terminer le programme (proprement).

⚠ Une valeur de saturation trop élevée ne doit pas empêcher votre watchdog de terminer le système.

Etape 3 : Fonctionnalité dégradée

Dans cette étape, vous devez modifier/utiliser le canari afin d'anticiper une surcharge en dégradant le traitement de la vidéo lorsqu'une surcharge survient (suffisament progressivement pour être anticipée). Documentez la méthode mise en oeuvre.

De plus, vous devrez trouver au moins 2 moyens de dégrader la tâche vidéo et les tester. Elaborez une stratégie pour appliquer l'un ou l'autre ou les deux afin de rendre votre système aussi résilient que possible tout en conservant le traitement le plus optimal possible. Une baisse de la surcharge devra donc permettre à votre système de revenir au traitement initial. Documentez le tout et incorporez la valeur de saturation maximale que votre stratégie vous permet d'atteindre avant que le watchdog n'arrête le système.

Travail à effectuer

A vous de développer le code correspondant aux différentes étapes. Seul le code final sera rendu et il devra donc être correctement commenté.

Un petit rapport présentant vos choix architecturaux ainsi que les tests effectués vous est également demandé.

Vous devez faire valider le fonctionnement de chaque étape auprès de l'assistant avant le rendu final.

- Rendez le tout sur cyberlearn, dans un fichier compressé nommé `rendu.tar.gz`
  - Ce fichier doit être généré en lançant le script `ptr_rendu.sh`
  - Le script vérifie qu'un fichier `rapport.pdf` est présent à son niveau, ainsi qu'un dossier `code` également présent au même niveau