HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Reconfigurable & embedded Digital Systems

Prof. Yann Thoma

Laboratoire de Programmation Temps Réel

semestre automne 2020 - 2021

Laboratoire 3 : Linux et Xenomai

Temps à disposition : 4 périodes

Objectifs

Ce laboratoire va vous permettre d'expérimenter les différences entre threads temps réel et non temps réel.

Étape 1 : Priorité des threads et Linux

Lors du dernier labo vous avez testé les capacités d'une application à servir des tâches périodiques sous différentes conditions. Nous allons maintenant effectuer des tests et récolter des données dans un environnement plus chargé en termes d'utilisation du temps processeur. Afin de créer un tel environnement, nous allons utiliser les commandes suivantes :

```
# taskset -pc 0 $$
# mkdir /home/redsuser/YYY/
# sudo /usr/lib/xenomai/testsuite/dohell -b /home/redsuser/cpu_loop \
       -s 10.192.XXX.XXX -m /home/redsuser/YYY 70
```

À vous de choisir le répertoire temporaire `YYY` que vous désirez utiliser.

Les options du script `dohell` sont les suivantes :

1. -b *hackbench* : chemin de l'application ou du script à stresser
2. -s *server* [-p *port*] : destination à laquelle des paquets réseau seront envoyés via `nc` (ou `netcat`) ; le port par défaut est 9
3. -m *folder* : répertoire (ou point de montage) dans lequel un fichier de 100MB sera créé et rempli de 0 grâce à une boucle infinie
4. *duration* : temps d'exécution du script, en secondes

Collectez les statistiques en exécutant l'application que vous avez écrite durant le laboratoire précédent pendant plus d'une minute et comparez vos résultats avec ceux du laboratoire précédent.

L'adresse IP (XXX.XXX) sera donnée en classe, et n'oubliez pas d'exécuter la même commande `taskset` si vous utilisez deux terminaux.

Étape 2 : Priorités

✠ Maintenant que nous avons un environnement pour lancer les expériences, nous allons l'utiliser sur différentes solutions. Ecrivez un programme qui exécute une tâche périodique dans une tâche haute priorité et qui utilise la fonction `nanosleep()` pour attendre une période de temps à l'intérieur d'une boucle.

Utilisez le même environnement et `dohell` pour collecter les mêmes statistiques. Comparez cette nouvelle solution en termes de données statistiques avec la solution précédente.

Étape 3 : Xenomai

Nous allons maintenant exploiter des tâches Xenomai via l'interface native de Xenomai. Ecrivez une nouvelle version de votre tâche périodique dans un fichier que vous nommerez `xenomai_timer.c`. Ensuite, effectuez les mêmes tests de l'étape 1 avec `dohell` et comparez vos résultats.

La documentation de l'API Xenomai est disponible ici.

Quelques points pour la mise au point de votre premier programme Xenomai :

1. Vous aurez besoin des fichiers d'entête suivants :

   ```
   #include <rtdk.h>
   #include <native/task.h>
   #include <native/timer.h>
   ```

2. Au début de votre programme, utilisez la fonction `mlockall(MCL_CURRENT|MCL_FUTURE)`. Il s'agit d'une fonction offerte par Linux. Trouvez en quoi elle sera utile dans un contexte temps-réel.
3. Un descripteur de tâche est une variable de type `RT_TASK`
4. Pour lancer une nouvelle tâche, utilisez les fonction `rt_task_create()` et `rt_task_start()` ou une seule fonction : `rt_task_spawn()`
5. La fonction `rt_timer_read()` vous permet de récupérer la valeur du timer
6. Pour mettre en place une tâche périodique, la fonction `rt_task_set_periodic()` vous permet de définir une période, puis la fonction `rt_task_wait_period()` permet d'attendre la prochaine requête périodique
   — Pour la définition de la périodicité, utilisez `rt_task_self()` pour récupérer le descripteur de la tâche courante, et `TM_NOW` pour lancer directement le timer
7. Pour l'affichage, l'usage de la fonction `printf()` pose problème dans un contexte temps réel. Pour éviter les problèmes y relatifs, utilisez la fonction `rt_printf()`, avec le même prototype. Il faudra juste exécuter l'appel suivant au début de votre programme : `rt_print_auto_init(1);`

Lorsque vous aurez écrit votre programme, utilisez le Makefile fourni pour le compiler. Il fixe certaines configurations pour l'utilisation de Xenomai. Mettez le Makefile dans le même dossier que vos sources, et lancez la commande suivante :

```
# make
```

Pour supprimer les fichiers générés, utilisez la commande suivante :

```
# make clean
```

✓✠ Commentez votre code et les résultats obtenus.

Travail à effectuer

Un rapport au format PDF devra contenir :

1. le code produit durant le laboratoire (indiqué par le symbole ✠), avec commentaires et explications nécessaires
2. les réponses aux questions directes (indiqué par le symbole ✓)

— Rendez le rapport et le code 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

Les connaissances acquises durant les laboratoires seront également testées lors des tests écrits ainsi que lors de l'examen final.