S. Lavirotte
Polytech Nice Sophia
SI4
2020-2021

TD n° 10
Introduction OS Temps Réel

Pour ce TD, nous allons mettre en évidence l'apport d'une approche temps réel à un noyau comme Linux. Nous allons faire une mise en œuvre du patch Linux `PREEMPT_RT` qui, s'il ne fournit pas un noyau temps réel dur, améliore largement le comportement du système lors de fortes charges.

1 Mesures sur un noyau avec Preempt par défaut du noyau Linux

Pour pouvoir réaliser une comparaison, la première chose est d'effectuer des mesures sur votre noyau actuel qui est issu de la branche principale donc avec une gestion de `PREEMPT` classique vous Linux (les tâches utilisateurs sont préemptibles, mais pas le noyau lui-même).

Vous allez donc ajouter à votre système un ensemble d'utilitaires qui vont vous permettre de mesurer la qualité d'un système temps réel et pour « charger » votre système et voir comment celui-ci réagit.

1.1 Installation d'outils de mesure de la qualité d'un système temps-réel

Il existe de nombreux outils pour mesurer la qualité d'un système temps réel. Certains s'intéressent aux temps de commutation entre tâches, d'autres à la précision des timers, à la latence des interruptions, aux temps de prise d'un mutex, etc.

Pour ce TD, j'ai retenu d'utiliser un outil dont les résultats sont assez représentatifs du comportement temps réel global d'un système : `cyclictest`. Initialement écrit par Thomas Gleixner, ce programme est maintenant intégré dans la suite `rt-tests` maintenue par Clark Williams.

Le principe de `cyclictest` est de vérifier la précision des déclenchements de tâches périodiques. Il programme une tâche qui doit être réveillée toutes les millisecondes. Lors de son activation, elle vérifie l'heure système et compare le temps écoulé depuis le dernier réveil et la période prévue. Après un nombre conséquent de déclenchements, on a un bon aperçu du comportement temps réel du système pour ce qui concerne un traitement périodique.

Pour vous éviter de télécharger et cross-compiler ces outils, nous avons intégrés ceux-ci à une image disque qu'il va falloir ajouter à votre machine virtuelle.

```
http://trolen.polytech.unice.fr/cours/isle/td10/sdg-rt_linux.7z
```

Monter ce nouveau disque en `/work/td10` (vous aurez aussi besoin du disque avec Buildroot à monter en `/work/td05`).

N'oubliez pas de faire vos modifications sous Buildroot en tant que `user` et pas `root`.

Intégrer à votre image système pour la Raspberry Pi toute l'arborescence qui se trouve dans le dossier `/work/td10/install/*`. Nous vous rappelons que pour intégrer cela proprement à votre système construit par Buildroot, il faut utiliser le dossier `overlay` puis recompiler Buildroot pour produire l'image qui contient le nouvel overlay). Par exemple :

```
cp -ar /work/td10/install/* /work/td05/raspberrypi3/overlay/
```

Il ne vous reste alors qu'à recompiler Builroot pour regénérer la nouvelle image incluant ces exécutables.

1.2 Utilisation des outils de mesure

Si vous avez été un peu curieux avant de faire la copie comme indiqué précédemment, vous aurez noté entre autres qu'un dossier `/usr/local/bin` contient des exécutables qui permettent de stresser le système à tester.

Vous commencerez par vous connecter à votre Raspberry Pi 3 en tant que super utilisateur (afin d'avoir un accès vous permettant de changer la politique d'ordonnancement comme l'utilise le programme que nous allons lancer).

```
ssh root@raspberrypi.local  (ou avec l'adresse IP sur votre réseau)
```

TD n° 10
Introduction OS Temps Réel

Vous veillerez à modifier votre PATH avec la commande suivante :

```
export PATH=$PATH:/usr/local/bin
```

Vous exécuterez alors le script `stress-test.sh` qui produira deux fichiers de log (`log1.txt` et `log2.txt`) qui correspondent aux temps de latence mesurés à l'aide du programme `cyclictest`.

Une fois les fichiers créés, vous pourrez lancer les commandes suivantes pour extraire les valeurs numériques qui nous intéressent et que nous allons exploiter.

```
cat log1.txt | grep "^0" | cut -d ' ' -f 2 > log1-data.txt
cat log2.txt | grep "^0" | cut -d ' ' -f 2 > log2-data.txt
```

Il vous faudra récupérer ces fichiers de log (les 4 pour bien comprendre) sur votre machine de travail pour copier les résultats dans un tableur afin de réaliser un graphique qui nous permettra d'analyser les résultats obtenus.

Vous copierez donc les valeurs dans les fichiers `log*-data.txt` dans le tableur (Excel) que vous récupérez à l'adresse suivante :

```
http://trolen.polytech.unice.fr/cours/isle/td10/Analyse-PREEMPT_RT.xlsx
```

Ceci vous permettra de comparer les temps de latence sans charge du système et avec un système chargé à `100%`. Que constatez-vous ? Quelle conclusion pouvez-vous tirer de cette première version du graphique ?

2 Mesures sur un noyau avec PREEMPT_RT

Pour voir le comportement du système avec la prise en compte des aspects temps réel dans le noyau Linux, nous allons appliquer le patch `PREEMPT_RT` sur les sources su noyau.

2.1 Création du noyau Linux avec PREEMPT_RT

Vous téléchargerez donc le patch pour la version du noyau que vous utilisez dans le système que vous construisez. Pour le vérifier, vous devez vous rendre dans le dossier `buildroot/output/build/linux...` et lancer la commande `make kernelversion`.

```
# make kernelversion
4.19.97
```

Vous téléchargerez alors le patch correspondant à votre version de noyau et l'appliquerez aux sources du noyau :

```
wget https://cdn.kernel.org/pub/linux/kernel/projects/rt/4.19/older/patch-4.19.94-rt39.patch.gz
gunzip -c patch-4.19.94-rt39.patch.gz | patch -p1
```

Vous relancerez alors la configuration du noyau (depuis le répertoire Buildroot, lancer `make linux-menuconfig`) pour activer les fonctionnalités de `PREEMPT_RT` et vous recompilerez votre noyau.

Au lieu d'aller prendre un café en attendant la fin de la compilation du noyau avec ce nouveau patch (et ça va être long car il y a un paquet de modules), vous êtes invités à aller lire le très bon document sur l'inversion de priorité dans la mission sur Mars Path Finder. Cela vous permettra sûrement de mieux comprendre les problématiques que nous sommes en train de mesurer (et d'avoir un aperçu/réviser la programmation concurrente).

```
http://wiki.csie.ncku.edu.tw/embedded/priority-inversion-on-Mars.pdf
```

Après compilation, une nouvelle `sdcard.img` est produite, incluant la fonctionnalité temps-réel au noyau. Nous allons ainsi pouvoir constater si le temps de réponse même à très forte charge est mieux respectée.

TD n° 10
Introduction OS Temps Réel

2.2 Mesure des temps de latence avec Linux PREEMPT_RT

Vous reprendrez la section précédente pour générer les deux fichiers de log avec les mesures cette fois-ci sur un système Linux temps réel. Après avoir copié les informations dans le fichier Excel, vous comparerez les mesures obtenues avec le système avec `PREEMPT_RT` et le système sans.

3 Conclusion

Quelles conclusions pouvez-vous tirer ? Que constatez-vous quant au temps maximum observé pour la réponse du système dans les deux cas ?

D'après les informations sont vous disposez grâce au cours, est-il possible d'aller plus loin dans cette direction (avec un noyau Linux) ? Si oui, comment ? Si non, ou s'il y a rapidement une limitation, quelle direction faut-il prendre pour disposer d'un système temps réel ? Argumentez.