S. Lavirotte
Polytech'Nice – Sophia
SI4
2018-2019

TD n° 7
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/td07/sde-rt_linux.7z
```

Monter ce nouveau disque en `/work/td07` (vous aurez aussi besoin du disque de la semaine dernière avec Buildroot à monter en `/work/td06`).

Intégrer à votre image système pour la Raspberry Pi tous les exécutables qui se trouvent dans le dossier `/work/td07/install`. Nous vous rappelons que pour intégrer cela proprement à votre système construit par Buildroot, il faut utiliser le dossier `overlay`).

1.2 Utilisation des outils de mesure

Vous disposez d'un script dans `/work/td07/stress-test.sh` qui produira deux fichiers de log (`log1.txt` et `log2.txt`) qui correspondent aux temps de latence mesurés.

Pour récupérer ces fichiers sur votre machine de travail, il vous faudra copier ces fichiers sur la partition `fat` ou `ext` correspondant à votre système natif pour vous faciliter l'extraction de ces fichiers de la carte SD. Donc si vous êtes sous un Linux sur votre machine natif de travail, vous pouvez laisser les fichiers dans le dossier où ils se trouvent (`/root`). Si vous avez un système Windows natif, il vous faudra monter la partition `1` de votre carte SD pour y copier les fichiers de log produits

```
mount /dev/mmcblk0p1 /mnt
cp log*.txt /mnt
umount /mnt
```

TD n° 7
Introduction OS Temps Réel

Depuis votre machine de travail (vous récupérerez les fichiers de votre machine native à votre machine de travail via le dossier de partage habituel), vous extrairez ensuite de chacun de ces deux fichiers la deuxième colonne des valeurs produites.

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

Vous copierez ces valeurs dans le fichier Excel que vous récupérez à l'adresse suivante :

```
http://trolen.polytech.unice.fr/cours/isle/td07/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
4.9.52
```

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

```
https://cdn.kernel.org/pub/linux/kernel/projects/rt/4.9/older/patch-4.9.47-rt37.patch.gz
```

Vous relancerez alors la configuration du noyau 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, 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.

```
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.

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

Quelle conclusion 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.