HEIG VD | REDS Institut Reconfigurable and Embedded Digital Systems

Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Alexandre Corbaz, Fiorenzo Gamba
Assistants : Anthony Jaccard, Basile Cuneo, Jean-Pierre Miceli

Ordonnancement

lab09 (Semaine du 27.11.2023)

Objectifs de laboratoire

Ce laboratoire porte sur les notions d'ordonnancement, en particulier la gestion de la priorité au niveau des processus

Récupération et rendu du laboratoire

Pour récupérer la branche de ce laboratoire, utilisez les commandes:

1. "*git fetch upstream*"
2. "*git checkout labXX"*

Ce laboratoire sera à rendre selon la méthode vue lors du précédent laboratoire, à savoir

1. Ajouter les fichiers modifiés avec "*git add <vos_fichiers>"*
2. Commiter les changements avec "*git commit –m "<message de commit>"*
3. Envoyer les changements sur votre repo gitlab avec "*git push –u origin"* ou "*git push —set-upstream origin"*

Etape 1 – Ajout de l'appel système « *renice()* » et gestion des priorités

Nous avons modifié *SO3* pour qu'il utilise une politique d'ordonnancement par priorité statique. L'implémentation de cet algorithme est disponible dans le fichier « *so3/kernel/schedule.c* ».

Exécuter ces commandes pour recompiler complètement SO3 avec sa nouvelle configuration :

```
reds@reds2022:~/sye23_student$ make clean
reds@reds2023:~/sye23_student$ make
```

Lorsqu'un processus est créé, la priorité par défaut (**10**) lui est assignée. La commande du shell « ***renice*** » doit permettre de changer à la volée cette priorité. Selon la convention vue en cours, plus la valeur numérique de la priorité est haute, plus le processus est prioritaire.

L'appel système « ***renice()*** » a été rajouté, mais sans implémentation. L'application « ***time_loop*** » sera utilisée dans les différents tests. Cette application permet d'afficher un compteur toutes les secondes pendant 30 itérations. Elle peut se lancer de deux façons différentes :

- Mode « **bloquant** » (***time_loop b***) : L'attente de la seconde se fait via **usleep**.
- Mode « **continu** » (***time_loop***) : L'attente se fait via une boucle « while » qui va lire le temps en continu et en sortir à chaque fois que le temps récupéré est plus grand d'une seconde.

⇒ Il est utile de retenir que le **shell** a le PID **1**.

a) Compléter l'implémentation de **« *renice()* »** dans le fichier « *so3/kernel/process.c* » (noyau) afin de pouvoir changer la priorité des processus.

b) Lancer l'application « ***time_loop*** » en mode **continu**, en arrière-plan (avec le symbole &) depuis le *shell*. Essayer d'exécuter des commandes dans le shell.

⇒ Que se passe-t-il et pourquoi ? (Répondre dans le fichier « ***rapport.md*** »)

c) Changer la priorité du *shell* pour que le processus « ***time_loop*** » devienne plus prioritaire. Essayer d'éxecuter quelques commandes dans le shell.

⇒ Que se passe-t-il et pourquoi ? (Répondre dans le fichier « ***rapport.md*** »)

d) Refaire la manipulation en mode **bloquant,** toujours en lançant l'application en background.

Comparer les deux modes et expliquer leurs différences de comportement (Répondre dans le fichier « ***rapport.md*** »).
Vous pouvez continuer à explorer les différents scénarios créés par l'ordonnancement par priorité statique en lançant par exemple deux processus « ***time_loop*** » et en jouant avec leur priorité respective.

Pour le prochain laboratoire, n'oubliez pas de recompiler SO3 avec

```
reds@reds2022:~/sye23_student$ make clean
reds@reds2023:~/sye23_student$ make
```

Pour disposer à nouveau du scheduling par round robin