Cours "Informatique Embarquée"

François Armand

M2 (IMPAIRS, LP, MIC, EIDD...)

Exercice N°5, 25 Novembre 2017

A rendre le 07/12/2017 Minuit

armand@informatique.univ-paris-diderot.fr

1. Inversion de priorité
   Le but de ce TP est de construire un programme de test automatique qui détermine s'il est possible de créer une situation d'inversion de priorité entre 3 threads. Le programme doit simplement imprimer une chaîne « Inversion détectée » ou « Inversion non détectée ».
   Il est recommandé de lire l'énoncé jusqu'à la fin avant de commencer.

   On va procéder par étapes pour y parvenir.

   1.1. Il faut évidemment créer 3 threads A, B et C. Chacune aura une priorité fixe différente. Par exemple A aura la plus haute priorité, C la plus faible, et B une priorité intermédiaire. Ces 3 threads devront donc utiliser une politique d'ordonnancement de type priorité fixe FIFO.

   1.1.1. Attribuer une priorité fixe pour un ordonnancement de type temps-réel, requiert des privilèges. On utilisera donc QEMU ou une machine virtuelle Linux, pour ce faire. Si vous n'avez pas fini le TP vous donnant accès à ces outils, utilisez une machine virtuelle de votre choix. En cas de problème, demandez de l'aide.

   1.1.2. Il est possible de fixer ces paramètres lors de la création des threads. Il est aussi possible de changer ces paramètres après création des threads, mais cela complique un peu la tâche.

   1.1.3. Pour des raisons de simplicité / symétrie on créera 3 threads en plus de la thread principale.

   1.2. Il faut ensuite que la thread la plus prioritaire et la thread la moins prioritaire rentrent en concurrence sur un objet de synchronisation (mutex). Il faut évidemment s'assurer que la thread la moins prioritaire obtient ce verrou avant que la thread la plus prioritaire tente d'acquérir à son tour ce verrou. Il vous faut donc trouver un moyen qui assure **avec certitude** que les acquisitions de verrou se déroulent dans l'ordre requis.

   1.3. Quand la thread de haute priorité se bloque sur le verrou déjà acquis par la thread de basse priorité, il suffit alors que la thread de moyenne priorité s'exécute en préemptant la thread la moins prioritaire pour créer cette situation d'inversion de priorité. Là encore, il vous suffit de trouver un moyen pour que cette thread de priorité intermédiaire démarre au bon moment.

   1.4. On décrétera que l'on a rencontré une inversion de priorité si la thread de priorité intermédiaire réussit à s'exécuter. On dira qu'il n'y a pas eu d'inversion de priorité si la thread de basse priorité peut relâcher le verrou convoité par la thread de haute priorité avant que la thread de moyenne priorité ait pu s'exécuter.

   1.5. Il est conseillé de faire un dessin (chronogramme) représentant l'exécution des threads à travers le temps, et d'y représenter schématiquement les opérations d'acquisition de verrou et de synchronisation entre les threads. Une fois le dessin fait et les événements de synchronisation représentés, la programmation finale devrait couler de source. Si, si!

2. Vérification du programme

Normalement, dans l'environnement Linux en classe d'ordonnancement FIFO, le programme devrait systématiquement détecter une situation d'inversion de priorité. On modifiera le programme pour remédier à cette situation d'inversion de priorité. En fonction de la présence d'un argument ou pas sur la ligne de commande, ce test n'utilisera pas l'héritage de priorité et détectera l'inversion de priorité, ou utilisera un mécanisme d'héritage de priorité et ne détectera pas d'inversion. Il sera utile de définir la macro suivante: #define _GNU_SOURCE