# Cours "Informatique Embarquée"

François Armand

## M2 SRI/Crypto

## Exer i e !" #$ % &é embre 2'()

## A rendre *e 2)/(2/2'(2 2+ , %- ./0

armand 1 informatique2uni3/paris/diderot2fr

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 !usqu'" la #in avant de commencer.

$n va procéder par étapes pour % parvenir.

1.1. Il #aut évidemment créer 3 threads &' ( et ). )hacune aura une priorité #i*e di##érente. Par e*emple & aura la plus haute priorité' ) la plus #aible' et ( une priorité intermédiaire. )es 3 threads devront donc utiliser une politique d'ordonnancement de t%pe priorité #i*e +I+$.

(2(2(2Attribuer une priorité fixe pour un ordonnan ement de type temps/rée*$ requiert des pri3i*45es2 6n uti*isera don 7EM8 9ou : ; M0$ pour e faire2 Si 3ous n<a3e= pas fini *e >? 3ous donnant a 4s @ es outi*s$ uti*ise= une ma Aine 3irtue**e de 3otre Aoix2 En as de prob*4me$ demande= de *<aide2

1.1.,.$n cherchera comment #aire. Il est possible de #i*er ces param-tres lors de la création des threads. Il est aussi possible de changer ces param-tres apr-s création des threads' mais cela complique un peu la t.che.

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

1.,. Il #aut ensuite que la thread la plus prioritaire et la thread la moins prioritaire rentrent en concurrence sur un ob!et de s%nchronisation 0mute*1. Il #aut évidemment s'assurer que la thread la moins prioritaire obtient ce verrou avant que la thread la plus prioritaire tente d'acquérir " son tour ce verrou. Il vous #aut donc trouver un mo%en qui assure avec certitude que les acquisitions de verrou se déroulent dans l'ordre requis.

1.3. 2uand la thread de haute priorité se bloque sur le verrou dé!" acquis par la thread de basse priorité' il su##it alors que la thread de mo%enne priorité s'e*écute en préemptant la thread la moins prioritaire pour créer cette situation d'inversion de priorité. L" encore' il vous su##it de trouver un mo%en pour que cette thread de priorité intermédiaire démarre au bon moment.

1.3. $n décrétera que l'on a rencontré une inversion de priorité si la thread de priorité intermédiaire réussit " s'e*écuter. $n dira qu'il n'% a pas eu d'inversion de priorité si la thread de basse priorité peut rel.cher le verrou convoité par la thread de haute priorité avant que la thread de mo%enne priorité ait pu s'e*écuter.

1.4. Il est conseillé de #aire un dessin 0chronogramme1 représentant l'e*écution des threads " trori `a1±%Mrede rac

,. 8éri#ication du programme

9ormalement' dans l'environnement Linu* en classe d'ordonnancement +I+$' le programme devrait s%stématiquement détecter une situation d'inversion de priorité. $n modi#iera le programme pour remédier '' cette situation d'inversion de priorité. :n #onction de la présence d'un argument ou pas sur la ligne de commande' ce test n'utilisera pas l'héritage de priorité et détectera l'inversion de priorité' ou utilisera un mécanisme d'héritage de priorité et ne détectera pas d'inversion. Il sera utile de dé#inir la macro suivante; <de#ine =>9 5=6$ 5 ? ) :