# Cours "Informatique Embarquée"

**François Armand**

## M2 IMPAIRS/MIC/LP/EIDD...

## Exercice N° 4, 3 Novembre 2017

## A rendre avant le 16/11/2017 minuit

**armand@informatique.univ-paris-diderot.fr**

# Performances et Benchmark

## 1. Consignes :

Habituelles. Cf Site du cours. (Temps, relecture, PDF, Tar.gz, Linux...)

## 2. Mesures de performances

Le but de cet exercice est d'effectuer :

- Une mesure de performance comparative entre la création/terminaison/notification de terminaison d'un processus et la création/terminaison/notification de terminaison d'une thread. On portera autant d'attention à la manière d'effectuer la mesure qu'au résultat de cette mesure.
- Une mesure de performance comparative entre le changement de contexte (context switch) entre 2 processus et le changement de contexte entre 2 threads appartenant au même processus.
- Une mesure de performance du coût d'allocation mémoire.
- De comparer avec d'autres valeurs, obtenues par un programme de mesures autre (lmbench, ou autre si vous trouvez d'autres)
- De comparer avec les valeurs obtenues par un autre (groupe d') étudiant(e-s)

## 1. Etapes pour la création/destruction

### 1.1. Quel outil pour quelle mesure?

Si l'on a un mètre avec seulement des graduations en centimètres à notre disposition, et que l'on veuille mesurer l'épaisseur d'une feuille de papier issue de ramette(s) également à notre disposition, comment procédera-t-on?

Cet exemple a-t-il un rapport avec la question posée dans ce TP? Pourquoi?

### 1.2. Question optionnelle : Identifiez les fonctions de mesure de temps disponibles dans un environnement Linux.

On se servira, si nécessaire, des commandes « `apropos` », « `man` » ou de recherche sur le web.
Pour chacune des fonctions trouvées, on indiquera:

- s'il s'agit d'un appel système (section 2 du manuel) ou d'une bibliothèque (section 3 du manuel)
- l'unité de temps utilisée par cette fonction,
- la précision de la mesure permise par cette fonction sur le système où l'on réalisera les mesures de performance. Il vous est donc demandé d'établir quel est le plus petit intervalle de temps observable au moyen des fonctions que vous aurez identifiées. Cet intervalle n'est pas forcément l'unité de la valeur renvoyée par la fonction. Une mesure effectuée par un décamètre pourrait être exprimée en mètres, il n'en reste pas moins qu'un décamètre ne peut pas mesurer une longueur avec une précision inférieure à 10 mètres.

- Pour les fonctions renvoyant une durée écoulée depuis un moment dans le passé, vous tenterez de définir/trouver ce point dans le passé. Vous tenterez de vérifier si la valeur renvoyée peut « déborder ».

1.1. **Écrivez un programme (en C) qui mesure le temps d'exécution des opérations suivantes :**

- Création et attente de la terminaison d'un processus dont la seule fonction est de se terminer immédiatement,
- Création et attente de la terminaison d'une « thread » dont la seule fonction est de se terminer immédiatement.
- La mesure réalisée indiquera le nombre de nanosecondes/µsecondes /millisecondes/secondes (selon l'unité utilisée par la fonction de mesure du temps que vous aurez utilisée) nécessaires, en moyenne, pour effectuer chacune des opérations ci-dessus. On pourra éventuellement fournir, un temps minimum et un temps maximum.
- Pour les 2 opérations mesurées (création de processus et création de thread), décrivez précisément ce que vous mesurez:
  - Quel est le point initial de mesure du temps,
  - Quel est le point final de mesure du temps
  - Quelle séquence d'opérations est ainsi mesurée. Est-ce bien ce que vous vouliez mesurer pour satisfaire à la demande faite dans ce sujet?
  - Commentez les résultats obtenus.

1.1.**Répétez votre mesure. Plusieurs fois.**
Le résultat obtenu est-il constant? Quelles méthodes statistiques devraient être utilisées pour « publier » des résultats ?

1.2. **Énumérez les phénomènes, facteurs qui peuvent influencer la mesure effectuée.**
Tentez d'être exhaustifs! Donnez des descriptions de manipulations permettant de mettre en œuvre ces facteurs et d'observer leur impact sur vos mesures.
Les résultats que vous avez obtenus, vous permettent-ils de fournir un temps minimum, et un temps maximum? Seriez-vous prêts à garantir ces indications de temps pour une utilisation dans un système critique?

# 5. Changement de contexte

La première chose à faire est donc de construire un programme qui mette en œuvre ces changements de contextes.

Pour la mesure entre 2 processus, il faut donc créer deux processus, qui s'exécuteront à tour de rôle. Comme le temps de changement de contexte est probablement faible, il est préférable de mesurer le temps d'un nombre important de changements de contexte. Pour se faciliter la vie, il est aussi préférable que les mesures de temps initial et final soient faites dans le même processus. Enfin, pour ne pas perturber la mesure, on s'assurera que chacun des 2 processus ne fait aucun traitement lorsqu'il est actif, sa seule activité étant de rendre la main au processus précédemment actif.

Il y a différentes manières de parvenir à un ordonnancement où l'on a successivement P1 et P2 (les 2 processus) qui sont actifs: P1, P2, P1, P2, P1, P2.... On peut se baser sur des mécanismes de synchronisation explicite :

- Les mécanismes de synchronisation explicites peuvent inclure l'utilisation de sémaphores, de tubes, de signaux…
- Les mécanismes implicites (sched_yield)

Pour la mesure entre les threads le principe est exactement identique.

Comme il existe sous Linux plusieurs classes d'ordonnancement, on fera les mesures suivantes :

1. Classe ordonnancement temps partagé :
    1. Changement de contexte explicite entre processus
    2. Changement de contexte explicite entre threads
2. Classe ordonnancement FIFO :
    1. Changement de contexte explicite entre processus
    2. Changement de contexte implicite entre processus
    3. Changement de contexte explicite entre threads
    4. Changement de contexte implicite entre threads

ATTENTION ! L'accès à la classe d'ordonnancement FIFO, nécessite d'avoir les droits super-utilisateur… Il faudra donc vous servir d'un système Linux sur votre machine personnelle, ou sur une VM, ou sur qemu… Les chiffres les plus intéressants étant évidemment obtenus sur une machine physique. Le but du TP n'étant pas les temps mesurés eux-mêmes, mais la manière de les obtenir, cela a peu d'importance.

Vous ferez attention aux architectures matérielles modernes des machines (multi-processeurs, multi-coeurs, hyper-threading…)

Dans votre compte-rendu, vous:

- Donnerez les résultats de vos mesures,
- Comparerez ces résultats et expliquerez les origines possibles des différences observées,
- Indiquerez brièvement comment vous avez procédé pour obtenir l'ordonnancement désiré, et en quoi votre choix peut avoir influé sur les résultats...
- Décrirez ce qui peut perturber les mesures que vous avez effectuées

## 5. Comparaison avec d'autres résultats de TP

5.1. Vous échangerez vos programmes de benchmark avec un(e) autre étudiant(e) (groupe) et vérifierez si les résultats obtenus par vos programmes et ceux de l'autre étudiant(e) (groupe) sont cohérents. Si vous constatez une différence significative vous tenterez d'en trouver l'explication.

## 6. Outils de « benchmarking »

6.1. Vous trouverez dans l'archive suivante: `~armand/Downloads/lmbench-3.0-a9.tar.gz` un jeu de programmes de mesures de performance, connus sous le nom de **lmbench**. Vous installerez ce benchmark, et vous l'exécuterez sur votre système Linux.

6.2. Vous comparerez les résultats obtenus pour fork par lmbench et ceux obtenus par vos soins.

6.3. Idem pour le coût des changements de contextes.

6.4. Dans les 2 cas ci-dessus, si vous constatez des différences « significatives » vous tenterez, d'en trouver l'explication.

6.5. Pour les très courageux, lmbench ne fournit pas (à ma connaissance) de benchmark pour les threads… On pourrait donc tenter de développer une extension à lmbench, pour inclure ce genre de mesures… Yakafokon !

Lmbench est une suite développée en Open Source. On peut visiter le site web correspondant: http://sourceforge.net/projects/lmbench