Bonneville Geoffrey - TP3

Temps estimé : 8 heures Temps nécessaire : 9 heures

1.1 On mesure la taille de la pile de feuilles de papier. On divise ensuite le résultat par le nombre de feuilles de papier.

Cet exemple a un rapport avec ce tp, car les fonctions pour mesurer le temps ne nous donne pas forcément la précision que nous cherchons.

Pour les 2 opération mesurées (fork et pthread), on mesure bien uniquement le temps entre le début de cration, et la fin de l'attente de la terminaison du processus/thread.

1.2 Pour ce TP, j'ai choisi d'utiliser la fonction `int clock_getres(clockid_t clk_id, struct timespec *res);` avec CLOCK_REALTIME comme premier argument. Il s'agit d'un appel système. Cette fonction renvoie une structure timespec, permettant de récupérer un temps en nanosecondes.

Mesures fork et thread

```
[geoffrey@ES1-Archlinux src]$ ./bench
Benchmarking fork...
	Mean time elapsed after 100 iterations = 226us
Benchmarking pthread...
	Mean time elapsed after 100 iterations = 91us
```

Ces résultats sont une moyenne calculée après 100 mesures. Après plusieurs essais, il apparait que les résultats ne sont pas constants. Pour publier des résultats, il faudrait répéter les mesures de nombreuses fois, sur un processeur dont la fréquence est fixe, et supprimer tout processus inutile qui fonctionne sur l'ordinateur.

Les facteurs qui peuvents influencer sur les mesures peuvent être :

- choix du "CPUFreq Governor"
- un processeur différent est assigné pour un processus
- processus qui tournent en arrière plan
- Scheduler qui choisit des processus différents

Les résultats obtenus ne peuvent garantir que l'on obtient un minimum et un maximum que l'on de dépassera pas. Ces résultats ne peuvent être utilisé pour un système critique.

Changements de contexte

Les résultats obtenus pour une éxécution sont:

```
[geoffrey@ES1-Archlinux src]$ ./bench
```

```
Benchmarking context switches for fork...
	Time elapsed after 100 switches = 978us
Benchmarking context switches for pthread...
	Time elapsed after 100 switches = 704us
```

On pourrait retirer à la mesure des changements de contexte pour les threads le temps de création d'un thread. (voir le code)

Tests avec lmbench

temps de création d'un processus

```
[geoffrey@ES1-Archlinux x86_64-linux-gnu]$ ./lat_proc fork
Process fork+exit: 196.6000 microseconds
```

On obtient 196 microsecondes contre 226 microsecondes avec mon programme.

Explications possibles

lmbench s'assure peut-être que le programme s'éxécute dans un contexte répétable (éxécution avec la même priorité, réglages du scheduler etc...).