Systèmes Informatiques

TP6: processus et appels système

Attention: ce TP **sera** noté

1 Inspecter les appels système d'un programme

Le but de ce TP est de mesurer le coût d'exécution d'un appel système: combien de temps faut-il pour qu'un processus donne la main au système d'exploitation et que celui-ci rende la main au processus une fois l'appel système terminé?
Nous allons utiliser un petit programme `appel.c` que vous trouverez dans le `/share`. Ce programme ouvre le fichier `/dev/null`, crée une variable `buffer` d'une taille de 1 GB, et écrit cette variable sur le fichier `/dev/null`.

Information

Si vous n'avez jamais programmé en C, **pas de panique!** Nous n'allons vraiment rien programmer de compliqué.

- Pour compiler le fichier **appel.c** il faut taper la commande: **gcc appel.c**
  Le résultat est un fichier exécutable qui s'appelle **a.out**
- Pour exécuter le programme, il faut taper: **./a.out**
- Pour visualiser les appels systèmes effectués par le programme, il faut taper: **strace ./a.out**
- Pour mesurer le temps d'exécution du programme, il faut taper: **time ./a.out**
  (la valeur qui vous intéresse est la première: real)

Information

N'essayez pas de faire sur ce TP sur Windows, cela ne marchera pas. Il vous faut impérativement un système Unix: de préférence les machines des salles de TP, ou à la rigueur vous pouvez essayer sur des Mac. Si vous utilisez un Mac alors il faut remplacer la commande **strace** par **dtruss**.

Question 1: Compilez le programme et exécutez-le. Quel est ce fichier `/dev/null`? Que se passe-t-il quand on écrit dedans?

Nous voulons observer les appels système effectués par le programme. Lancez la commande **strace ./a.out**. Si le résultat déborde de votre écran vous pouvez utiliser la commande suivante pour écrire la liste des appels systèmes dans un fichier plutôt qu'à l'écran: **strace ./a.out 2> liste-des-appels.txt**

Question 2: Combien ce programme effectue-t-il d'appels système? Pouvez-vous retrouver ceux qui sont produits par votre programme (après le démarrage du programme)?

2 Mesurer le temps d'exécution d'un appel système

Pour mesurer le temps d'exécution d'un appel système, vous devrez modifier légèrement la fonction **ecrire()** de façon à modifier le nombre d'appels système tout en continuant d'écrire la même quantité de données dans **/dev/null**. Par exemple, au lieu d'écrire une fois 1 GB, vous pourriez faire deux appels qui chacun écrivent 0.5 GB ou 10 appels qui chacun écrivent 0.1 GB, etc. Il vous est fortement recommandé de définir le nombre d'appels sous forme d'une variable unique que vous pourrez ensuite modifier à votre guise.

Question 3: comment varie le temps d'exécution du programme lorsqu'on change le nombre d'appels système mais sans rien changer d'autre au comportement du programme? Il vous faudra tester des exécutions avec quelques millions d'appels système avant de constater une différence.

Question 4: tracez la courbe du temps d'exécution total par rapport au nombre d'appels système effectués. Pouvez-vous en déduire le temps d'exécution moyen d'un appel système?

Question 5: vérifiez la fréquence d'exécution du processeur de votre machine (il doit se situer aux alentours de 3 GHz). Quelle est la durée d'un cycle d'horloge? Combien de cycles d'horloge sont-ils nécessaires pour effectuer un appel système?

— fin —