Devoir final INF3173

Jean Privat

du 3 au 4 mai 2021

- Chaque devoir est **individuel**.
- Le sujet est en Markdown, répondez directement dans le fichier. Codage UTF-8, fins de lignes Unix.
- Répondez sur les lignes avec des chevrons (`>`), répondez après les chevrons ; vous pouvez en modifier le nombre, mais laissez les chevrons en place (pour que je voie vos réponses).
- Les limites maximales en nombre de mots des réponses seront interprétées avec intelligence.
- **Ne touchez pas** au reste du sujet. Un `diff` entre votre copie et le sujet ne **doit** faire apparaitre que vos réponses.
- Les réponses devront être aussi **détaillées** et **argumentées** que possible (en respectant la taille de la réponse attendue).
- Vous avez jusqu'au mardi 4 mai 2021 à 23h55 pour remettre votre fichier Markdown via Moodle. Vous pouvez faire autant de remises que vous le souhaitez, seule la dernière sera considérée.
- Merci de **ne pas discuter** du contenu du sujet sur Mattermost (ou via tout autre médium).
- Si vous avez des questions, vous pouvez toujours tenter de me les poser en message privé sur Mattermost, mais je me réserve le droit de **ne pas répondre** (ou de répondre après la date de remise).
- Tout non-respect du format ou des consignes sera pénalisé.
- La qualité du français ainsi que la rigueur, la précision et la justesse dans les explications et dans le vocabulaire du cours utilisé sera pris en compte.

Identification des étudiants

- Nom:
- Prénom:
- Code permanent:

Quiz

La première partie du devoir est sous forme d'un quiz indépendant. Quand le quiz est commencé, il doit être terminé en moins de 30 minutes.

- https://www.moodle2.uqam.ca/coursv3/mod/quiz/view.php?id=2342120

Programme forkos

Un programmeur débutant veut utiliser la programmation multiprocessus pour optimiser une application de calcul numérique haute performance. Pour ce faire, il développe le prototype `forkos.c` suivant qui implémente le calcul du cosinus dans un sous-processus afin d'être plus efficace.

```c
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
#include<math.h>

// Calcule le cosinus de x dans un sous-processus pour être plus efficace
double sub_cos(double x) {
    double cosix;
    int p[2];
    pipe(p);
    int pid = fork();
    if (pid>0) {
        write(p[1], &x, sizeof(x));
        read(p[0], &cosix, sizeof(cosix));
        close(p[0]);
    } else {
        read(p[0], &x, sizeof(x));
        cosix = cos(x);
        write(p[1], &cosix, sizeof(cosix));
        close(p[1]);
    }
    return cosix;
}

double getx(void) {
    double x;
    printf("x: ");
    fflush(stdout);
    fscanf(stdin, "%lf", &x);
    return x;
}

int main(void) {
    printf("%lf\n", sub_cos(getx()));
}
```

La compilation se passe bien (ne pas oublier l'option `-lm` qui permet la liaison

dynamique du programme avec la bibliothèque mathématique standard `libm.so`).

Première expérience

Le programmeur lance alors une première exécution. Il saisit 0 comme valeur de x. Le programme affiche la valeur 1, qui est correcte, mais semble l'afficher 2 fois. Puis, le programme se termine.

```
user@machine1$ ./forkos
x: 0
1.00000
1.00000
user@machine1$
```

Q01

Combien de processus différents sont en œuvre dans cette exécution du programme ?

(une ligne)

Q02

Pour chacun des processus de cette expérience, quels sont les appels système `read`, `write`, `pipe`, `close`, `fork`, `wait`, `exec`, `exit` effectués, dans quel ordre, et avec quels résultats ou effets ?

(pas de limite de place)

Q03

Pour chacun des processus de cette expérience, indiquez à quoi correspond chacun de leurs descripteurs de fichier. Indiquez à quel moment chacun est crée ou fermé.

(pas de limite de place)

Q04

Où est la bibliothèque mathématique lorsque le programme s'exécute ? Quelle est l'intrication entre la bibliothèque mathématique et chacun des processus du programme ? Que se passe-t-il pour libm lors du fork ?

(60 mots maximum)

Seconde expérience

Consciencieux, le programmeur débutant compile et exécute son programme forkos sur une autre machine et effectue la même expérience.

Cette fois-ci, l'affichage n'a lieu qu'une fois, mais le résultat affiché est mauvais, puis le programme se termine.

```
user@machine2$ ./forkos
x: 0
0.000000
user@machine2$
```

Par hasard, le programmeur effectue un `ps` et voit apparaitre, à sa grande surprise, son programme forkos dans la liste des processus.

```
user@machine2$ ps
    PID TTY          TIME CMD
1490052 pts/8    00:00:00 bash
2045081 pts/8    00:00:00 forkos
2045164 pts/8    00:00:00 ps
```

Q05

Pour chacun des processus de cette seconde expérience, quels sont les appels système `read`, `write`, `pipe`, `close`, `fork`, `wait`, `exec`, `exit` effectués, dans quel ordre, et avec quels résultats?

(pas de limite de place)

Q06

À quoi correspond le processus 2045081 de l'affichage du ps?

(une ligne maximum)

Q07

Dans quel état (actif, prêt, bloqué, zombi, etc.) est le processus 2045081?

(une ligne maximum)

Q08

Qu'est-ce que le processus 2045081 est en train de faire ou qu'attend-il? Comment pouvez-vous caractériser cette situation ?

(60 mots maximum)

Q09

Détaillez et expliquez une façon de faire faire afficher `1.000000` au processus 2045081 et de le terminer sans utiliser kill. (Indice: c'est possible de faire tout ça en une courte et seule ligne de shell)

(pas de limite de place)

Analyse

Q10

En s'adressant au programmeur débutant, expliquez les problèmes et défauts de son programme forkos et expliquez ce qui aurait dû être fait.

(pas de limite de place)

Q11

Appliquez vos corrections (cf Q10) au programme `forkos` et proposez ainsi une solution fonctionnelle. Vous devez conserver les choix de conception d'origine, en particulier le fork pour le calcul du cosinus dans un sous-processus et les communications par tubes pour les données et les résultats.

```
// TODO
```

Q12

Expliquez pourquoi l'approche initiale du programmeur est douteuse d'un point de vue de l'augmentation des performances; même en admettant que le programme soit débogué, cf questions Q10 et Q11, et que l'on calcule quelque chose de plus exigeant niveau CPU que le cosinus.

(pas de limite de place)

Q13 bonus

Quel est votre appel système préféré, et pourquoi ?

(pas de limite de place)