Devoir final INF3173

Jean Privat

du 15 au 16 mars 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 16 mars 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=2296701

Présentation du programme metsys

Un programmeur débutant développe le programme `metsys.c` qui permet d'exécuter indépendamment un programme ou une commande shell. L'astuce proposée par le programmeur est simple : si l'exécution directe du programme échoue, on l'exécute sous forme de commande shell grâce à l'option `-c` du shell.

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

int metsys(char **cmd) {
    pid_t pid = fork();
    if (pid > 0) {
        int res = execv(*cmd, cmd); // on tente d'exécuter sans le shell
        if (res == -1) {
            // si ça échoue, on réessaye, mais avec le shell cette fois!
            pid = fork();
            if (pid > 0) {
                execl("/bin/bash", "sh", "-c", *cmd, cmd, (char*)NULL);
            }
        }
    }

    int status=0;
    wait(&status);
    return status;
}

int main(int argc, char **argv) {
    int code = metsys(argv+1);
    printf("#status=%d\n", code);
}
```

Première expérience

Malheurusement, le programme ne fonctionne pas très bien.

Si on exécute `./metsys /bin/echo bonjour le monde`, on obtient

```
#status=0
bonjour le monde
```

Ce qui est surprenant, car l'affichage du `echo` apparait après la ligne de statut

(qui indique la valeur du code de retour) ce qui laisserait sous-entendre des entorses philosophiques au principe de causalité et à la théorie de la linéarité du temps.

Q1

Combien de processus sont mis en oeuvre lors de l'exécution de `./metsys /bin/echo bonjour le monde`?

Quels sont pour chacun d'eux :

- leurs relations de parenté ?
- les fork, exec, wait et exit réalisés ?
- leurs affichages ?

Détaillez et justifiez votre réponse.

(150 mots maximum)

Q2

Pourquoi l'affichage de la ligne de statut a lieu avant l'affichage du echo ? Est-ce systématique ou est-ce que l'ordre peut varier d'une exécution à l'autre ? Détaillez et justifiez votre réponse.

(150 mots maximum)

Q3

En prenant comme contexte l'exécution du programme dans cette première expérience, donnez un exemple précis et concret qui correspond à chacune des transitions d'états de processus suivantes:

Passage de l'état actif à l'état prêt.

Passage de l'état actif à l'état bloqué.

Passage de l'état actif à l'état zombi.

Seconde expérience

Cette fois-ci on exécute `./metsys echo au revoir`, mais cela affiche

```
#status=0
#status=0
```

Ce qui est très incorrect, car :

- « `au revoir` » n'est aucunement affiché
- Il y a 2 lignes de statut
- Le tout est terminé par une ligne blanche sans rapport

Q4

Combien de processus sont mis en oeuvre lors de l'exécution de `./metsys echo au revoir` ?

Quels sont pour chacun d'eux :

- leurs relations de parenté ?
- les fork, exec, wait et exit réalisés ?
- leurs affichages ?

Détaillez et justifiez votre réponse.

(150 mots maximum)

Q5 (question bonus difficile)

Pourquoi « au revoir » n'est pas affiché. Est-ce que cette exécution du programme aurait pu avoir un autre comportement ou un affichage différent ?

(200 mots maximum)

Critique et solution

Q6

En vous adressant directement au programmeur débutant, pointez chacun des nombreux problèmes de son programme et indiquez ce qui aurait dû être fait.

(200 mots maximum)

Q7

Appliquez vos corrections (cf Q6) au programme `metsys.c` et proposez ainsi une solution fonctionnelle.

`TODO`

Quiz inversé

Q8

Proposez une courte question **originale** de quiz sur les systèmes de fichiers (capsules 300 à 330) destinée à des étudiants du cours INF3173. La question doit être à choix de réponses, avec 4 réponses possibles, mais dont exactement une seule est correcte. Les 3 autres réponses doivent être suffisamment vraisemblables par rapport à la bonne réponse pour que la question soit intéressante et pertinente à l'évaluation des connaissances sur les principes des systèmes d'exploitation.

- Question:
- Mauvaise réponse 1:
- Mauvaise réponse 2:
- Mauvaise réponse 3:
- Bonne réponse: