Devoir intra INF3173

Jean Privat

du 27 octobre 2020 au 2 novembre 2020

- Chaque devoir est **individuel**.
- Le sujet est en Markdown, répondez directement dans le fichier. Codage UTF-8, fins de lignes Unix (`\n` seul).
- Répondez sur les lignes avec des chevrons (`>`), répondez après les chevrons et laissez les chevrons en place (pour que je vois vos réponses).
- Ne touchez pas au reste du sujet. Un `diff` entre votre copie et le sujet ne **doit** faire apparaitre que vos réponses.
- Les chevrons servent d'indicateur de la longueur attendu de la réponse. Chaque chevron correspond environ à 15 mots. Les limites maximales en mots seront interprétés avec intelligence.
- Les réponses devront être aussi argumentées que possible (en respectant la taille de la réponse attendue).
- Vous avez jusqu'au lundi 2 novembre 2020 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.

Identification des étudiants

- Nom:
- Prénom:
- Code permanent:

Questions de cours

Q1 Expliquez brièvement si l'instruction machine `call`, qui permet d'invoquer un sous-programme, a besoin du mode noyau pour être exécutée.

45 mots maximum.

Q2 Expliquez brièvement si un processus à l'état bloqué peut faire des appels système pour se débloquer.

45 mots maximum.

Q3 Expliquez brièvement si l'interprète de commande (*shell*) fonctionne en mode noyau.

60 mots maximum.

Q4 Expliquez brièvement si un processus doit gérer lui-même son entrée dans la table des processus.

60 mots maximum.

Retour sur le TP0

Dans le cadre de l'utilitaire `copie` du TP0.

Quand on exécute `strace ./copie -c -b 16 datam.bin tmp` où `datam.bin` est un fichier de 1 Mo (1048576 octets), on observe une longue séquence de `read` et de `write` alternés, de la forme :

```
read(3, "...", 4096) = 4096
write(4, "...", 4096) = 4096
read(3, "...", 4096) = 4096
write(4, "...", 4096) = 4096
```

Q5 Expliquez pourquoi `read` et `write` sont utilisés et que `fread` et `fwrite` n'apparaissent pas (alors que l'option`-c` est bien précisée).

60 mots maximum.

Q6 Expliquez pourquoi `read` et `write` utilisent des tailles de 4096 octets alors que l'option `-b` est précisée avec une valeur de 16.

Aide : vous pouvez lire la page de manuel `setvbuf(3)`.

60 mots maximum.

Q7 Expliquez pourquoi la variation `-l` (avec l'appel système `copy_file_range`) est beaucoup plus performante quand la taille devient grande.

60 mots maximum.

Ordonnancement

Soient 4 processus A, B, C, D avec les caractéristiques suivantes

Nom	temps d'arrivée	temps de calcul	échéance
A	0	6	9
B	1	4	12
C	2	1	13
D	3	2	5

Pour chacune des stratégies qui suivent (FIFO, RR, SJF et EDF):

- Complétez le diagramme de Gantt d'ordonnancement.
- Calculez le temps d'exécution moyen (temps de sortie - temps d'arrivée).
- Indiquez les tâches hors délai qui n'ont pas respecté leurs échéances (temps de sortie > échéance).

Q8 FIFO

Pour FIFO on vous donne le diagramme, ainsi que le format ASCII à utiliser: `-` pour prêt, `X` pour actif et espace pour absent (non démarré, terminé ou bloqué).

```
  0123456789012
A XXXXXX
B  -----XXXX
C   --------X
D    --------XX
```

- Temps d'exécution moyen:

- Tâches hors délai:

Q9 RR

Le quantum est de 2 unités de temps.

```
  0123456789012
A
```

```
B
C
D
```

- Temps d'exécution moyen:

- Tâches hors délai:

Q10 SJF (préemptif)

```
  0123456789012
A
B
C
D
```

- Temps d'exécution moyen:

- Tâches hors délai:

Q11 EDF

EDF (*earliest deadline first*) est un algorithme préemptif temps réel qui élit la tâche dont l'échéance est la plus proche.

```
  0123456789012
A
B
C
D
```

- Temps d'exécution moyen:

- Tâches hors délai:

Furk

Soit le programme C mystère et bogué suivant :

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

```c
#include<unistd.h>

/* PID des deux poussins. */
int p1;
int p2;

/* Crée 2 poussins.
 * Retourne 0 pour la maman, 1 pour le premier poussin et 2 pour le second poussin. */
int furk(void) {
    p1 = fork();
    p2 = fork();
    if (p1 == 0) return 1; // premier poussin
    if (p2 == 0) return 2; // second poussin
    return 0; // maman poule
}

int main(int argc, char **argv) {
    printf("Je suis %d\n", getpid());
    int p = furk();
    if (p == 0) printf("Mes poussins sont %d et %d\n", p1, p2);
    if (p == 1) printf("L'autre poussin est %d\n", p2);
    if (p == 2) printf("L'autre poussin est %d\n", p1);
}
```

Lorsqu'on l'exécute, celui-ci affiche quelque chose de potentiellement surprenant.

```
Je suis 1813665
Mes poussins sont 1813666 et 1813667
L'autre poussin est 1813666
L'autre poussin est 1813668
L'autre poussin est 0
```

Q12 Identifiez chacun des processus de ce programme, indiquez les liens de parenté.

Q13 Identifiez l'évolution des valeurs de `p1`, `p2` et `p` ainsi que les affichages de chacun des processus.

Q14 Expliquez le comportement observé et en quoi il diffère de comportement attendu du programmeur.

En particulier, pourquoi 5 lignes d'affichées, pourquoi le poussin 1813667 a disparu, et pourquoi il y a deux poussins inattendus 1813668 et 0.

150 mots maximum.

Exoc

Un programmeur débutant doit écrire un programme `exoc.c` pour lister le détail du contenu du répertoire de connexion de l'utilisateur. On lui donne une contrainte : il doit utiliser `execlp`.

```c
#include <stdlib.h>
#include <unistd.h>
int main(void) {
    execlp("cd", "cd ~", NULL);
    execlp("ls", "ls -l", NULL);
    return EXIT_SUCCESS;
}
```

À sa grande surprise, le programme compile et s'exécute sans planter. Par contre, il affiche le contenu du répertoire courant (au lieu du répertoire de connexion de l'utilisateur) et en version courte (comportement de `ls` par défaut) au lieu d'une liste détaillée (option `-l` de `ls`).

Q15 Sans juger de la qualité du programme, détaillez et expliquez le comportement observé.

90 mots maximum.

Q16 Jugez de la qualité du programme et, « en vous adressant » à ce programmeur débutant, expliquez les gros problèmes de son programme.

150 mots maximum.

Q17 Proposez une version correcte et courte de ce programme avec toujours la contrainte d'utiliser `execlp`.

Gérez, bien sûr, correctement les cas d'erreurs.

```c
/*TODO*/
```