Examen de programmation système

Juliusz Chroboczek

26 juin 2018

La durée de l'examen est de 2 heures. Les documents sont autorisés, le matériel électronique est interdit. Le sujet est composé de 3 pages.

Question 1. Indiquez le contenu du fichier « a » après l'exécution de chacun des fragments de code suivants. Si le programme se plante, il faudra inidiquer par quel signal il est tué.

1.
```
int fd;
fd = open("a", O_RDWR | O_CREAT | O_TRUNC, 0666);
ftruncate(fd, 8);
close(fd);
```
2.
```
int fd;
char *a;
fd = open("a", O_RDWR | O_CREAT | O_TRUNC, 0666);
ftruncate(fd, 8);
a = mmap(NULL, 8, PROT_READ | PROT_WRITE, MAP_PRIVATE, fd, 0);
*a = 65;
msync(a, 8, MS_SYNC | MS_INVALIDATE);
close(fd);
```
3.
```
int fd;
char *a;
fd = open("a", O_RDWR | O_CREAT | O_TRUNC, 0666);
ftruncate(fd, 8);
a = mmap(NULL, 8, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
*a = 65;
msync(a, 8, MS_SYNC | MS_INVALIDATE);
close(fd);
```
4.
```
int fd;
char *a;
fd = open("a", O_RDWR | O_CREAT | O_TRUNC, 0666);
ftruncate(fd, 8);
a = mmap(NULL, 8, PROT_READ, MAP_PRIVATE, fd, 0);
*a = 65;
msync(a, 8, MS_SYNC | MS_INVALIDATE);
close(fd);
```

Question 2. Bernard a écrit un programme qui fait huit gros calculs et affiche leur somme :

```
extern double groscalcul(int);

int main() {
    double s = 0.0;
    for(int i = 0; i < 8; i++)
        s = s + groscalcul(i);
    printf("lf\n", s);
    return 0;
}
```

Bernard est un peu embêté, car il a demandé à son patron de lui acheter une machine avec huit cœurs, et son programme est toujours aussi lent que sur l'ancienne, qui n'avait qu'un cœur. Alice lui explique que son programme est séquentiel : il s'exécute sur un seul cœur. Mais ça tombe bien, une réduction, ça se parallélise facilement.

1. Écrivez un programme C qui crée une zone de mémoire partagée *anonyme* (pas besoin de `open` ou `shm_open`) contenant un tableau `double valeurs[8]`. Il engendre ensuite huit processus numérotés de 0 à 7 ; le processus numéro i calcule `groscalcul(i)`, stocke le résultat dans `valeurs[i]`, puis termine. Le programme principal attend la mort de ses huit fils, puis affiche la somme des huit valeurs. (Vous n'êtes pas autorisés à utiliser des *threads*, et vous n'êtes pas autorisés à implémenter la communication entre les processus autrement qu'à l'aide de la zone de mémoire partagée.)
2. Expliquez *en deux phrases au plus* pourquoi il n'est pas nécessaire d'utiliser un sémaphore ou un *mutex* pour protéger la zone de mémoire partagée. (Indication : il y a deux raisons.)

Question 3. Bernard (toujours lui) a écrit un programme qui fait un gros calcul puis stocke le résultat dans un fichier :

```
extern double unautregroscalcul(void);

int main() {
    double v;
    FILE *f;
    v = unautregroscalcul();
    f = fopen("resultat.dat", "w");
    if(f == NULL) {
        perror("T'as perdu (et t'as un gros nez)");
        exit(1);
    }
    fprintf(f, "%lf\n", v);
    fclose(f);
    return 0;
}
```

Normalement, Bernard lance le calcul dans une fenêtre pendant qu'il joue au Sorceleur dans une autre. Cependant, il se trompe occasionnellement de fenêtre et appuie sur « `^C` » ce qui interromp le calcul.

Modifiez le code de Bernard pour que, lorsqu'on appuie sur « `^C` », le programme affiche « `T'as appuyé sur la mauvaise touche et t'as un gros nez.` » puis continue à s'exécuter (ne s'arrête pas). (Indication : on rappelle que la touche « `^C` » a pour effet d'envoyer le signal `SIGINT` au processus connecté au terminal.)

Question 4. À la réunion de leur équipe, Alice et Bernard présentent le code qu'ils ont écrit pour compter le nombre d'instances de l'entier 42 dans une (grande) matrice carrée. Alice a écrit le code de la figure 1. Par contre, Bernard a parcouru le tableau par colonnes, et a écrit le code de la figure 2.

```
int a[n][n];
...
int count = 0;
for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
        if(a[i][j] == 42)
            count++;
    }
}
```

Figure 1 — Le code d'Alice

```
int a[n][n];
...
int count = 0;
for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) {
        if(a[j][i] == 42)
            count++;
    }
}
```

Figure 2 — Le code de Bernard

Alice dit que son code est plus efficace, que c'est par lignes qu'il faut parcourir la matrice. Bernard dit que pas du tout, c'est par colonnes que ça ira plus vite. Chloé, qui a toujours été très forte en algorithmique, dit que ça n'a pas d'importance car les deux algorithmes sont en $O(n^2)$.

Dites qui a raison, et expliquez *en un paragraphe au plus* pourquoi. (Indication : en C, les tableaux sont stockés en mémoire par lignes, la case `a[i][j]` est suivie de la case `a[i][j+1]`.)