

Systèmes d'exploitation

Exercices accompagnant le cours : Processus (partie 1)

Objectifs des exercices

Approfondir les concepts de processus

1 Étude des processus en C

1- Ecrire un programme qui crée deux processus fils. L'un affiche les entiers de 1 à 50, l'autre de 51 à 100.

2- Indiquer ce qu'affiche le programme suivant :

```
int main() {
    int pid;
    int x = 1;

    pid = fork();
    if(pid == 0) {
        x = x + 1;
        printf("Dans_fils_:x=%d\n", x);
        exit(0);
    }

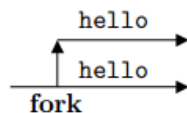
    x = x - 1;
    printf("Dans_pere_:x=%d\n", x);
    exit(0);
}
```

Que se passe-t-il si on enlève le premier exit(0); (dans le fils)?

3- Dessinez le schéma d'exécution des programmes suivants :

Exemple :

```
int main() {
    fork();
    printf("hello!\n");
    exit(0);
}
```



Questions :

```
int main() {
    fork();
    fork();
    printf("hello!\n");
    exit(0);
}
```

```
int main() {
    fork();
    fork();
    fork();
    printf("hello!\n");
    exit(0);
}
```

Combien de processus sont créés pour un nombre n de fork() effectués à la suite?

4- Combien de lignes "hello!" affiche chacun des programmes suivants?

Programme 1 :

```
int main() {
    int i;

    for(i=0; i<2; i++)
        fork();
    printf("hello!\n");
    exit(0);
}
```

Programme 2 :

```
void doit() {
    fork();
    fork();
    printf("hello!\n");
}

int main() {
    doit();
    printf("hello!\n");
    exit(0);
}
```

Programme 3 :

```
int main() {
    if(fork())
        fork();
    printf("hello!\n");
    exit(0);
}
```

Programme 4 :

```
int main() {
    if(fork()==0) {
        if(fork()) {
            printf("hello!\n");
        }
    }
}
```

2 Exécution d'un programme

La famille de primitives `exec` permet de créer un processus pour exécuter un programme déterminé (qui a auparavant été placé dans un fichier, sous forme binaire exécutable). On utilise en particulier `execvp` pour exécuter un programme en lui passant un tableau d'arguments. Le paramètre `filename` pointe vers le nom (absolu ou relatif) du fichier exécutable, `argv` vers le tableau contenant les arguments (terminé par `NULL`) qui sera passé à la fonction `main` du programme lancé. Par convention, le paramètre `argv[0]` contient le nom du fichier exécutable, les arguments suivants étant les paramètres successifs de la commande. Lorsque `execvp` est appelé, le processus est remplacé par le programme à exécuter. Il ne peut donc pas continuer son programme "prévu" après avoir exécuté `execvp`.

```
#include <unistd.h>
int execvp(char *filename, char *argv[]);
```

1- Que fait le programme suivant :

```
#include <stdio.h>
#include <unistd.h>
#define MAX 5
int main() {
    char *argv[MAX];
    argv[0] = "ls"; argv[1] = "-lR"; argv[2] = "/"; argv[3] = NULL;
    execvp("ls", argv);
}
```

3 Comportement de programmes

1- Décrivez le comportement de chacun de ces programmes suivant :

a:

```
int main() {
    main();
}
```

b:

```
int main() {
    while (1) fork();
}
```

c:

```
int main(int argc, char *argv[]){
    execvp(argv[0], argv);
}
```

d:

```
int main(int argc, char *argv[]){
    execvp(argv[1], argv);
}
```

2- Indiquer ce qu'affiche chacun des segments de programme suivants? Quel est le nombre de processus créés dans chaque cas?

a:

```
for (i=1; i<=4; i++) {
    pid = fork();
    if (pid != 0) printf("%d\n", pid);
}
```

b:

```
for (i=1; i<=4; i++) {
    pid = fork();
    if (pid == 0) break;
    else printf("%d\n", pid);
}
```

c:

```
for (i=0; i<3; i++) {
    p = fork();
    if (p < 0) exit(1);
    execlp("prog", "prog", NULL);
}
```

d:

```
for (i=1; i<=nb; i++) {
    p1 = fork();
    p2 = fork();
    if (p1 > 0) exit(0);
    if (p2 > 0) exit(0);
    execlp("prog1", "prog1", NULL);
    execlp("prog2", "prog2", NULL);
}
```

4 Bonus

4.1 Comportement de programmes

1- Décrivez le comportement de chacun de ces programmes suivant :

a:

```
int main( ) {  
    int p=1 ;  
    while(p>0)  
        p=fork() ;  
    execlp("prog", "prog", NULL) ;  
    return 0 ;  
}
```

b:

```
int i=4,  
int main ( ) {  
    j=10;  
    int p ;  
    p = fork();  
    if(p<0)  
        exit(1) ;  
    j += 2;  
    if (p == 0){  
        i *= 3;  
        j *= 3;  
    }  
    else {  
        j *= 2;  
    }  
    printf("i=%d, j=%d", i, j) ;  
    return 0 ;  
}
```

Remarque : Ce sujet a été composé à l'aide de nombreuses ressources disponibles sur la toile et plusieurs exercices sont inspirés d'exercices disponibles. La composition de ce sujet a été pensée dans un objectif purement pédagogique, et les auteurs respectifs gardent l'entière paternité de leur travail. Pour une liste exhaustive des ressources utilisées, se référer au cours.