## 1- Répondez aux questions suivantes, en considérant un contexte de système d’exploitation Linux :

### Qu’est-ce qu’un « Fork »?

|  |
| --- |
| Faire un fork consiste à diviser le programme en deux, en un père et en un fils. L’enfant sera exécuté, puis se sera le tour du parent. |

### Qu’est-ce qu’un « Thread »? *(Vous pouvez vous référer à « pthread »)*

|  |
| --- |
| Un thread est une tâche, ou un fil d’instructions. |

### Quelle est la différence entre les processus « Fork » et les processus « Thread » ?

|  |
| --- |
| Le fork multiplie par deux un processus tandis que le thread en crée un nouveau sans que celui-ci ne soit nécessairement une copie. |

### Donnez un exemple détaillé et commenté de démarrage d’un « Fork »

|  |
| --- |
| //Appel du fork  curStatus = fork()  //Si mon status == 0, alors je suis l’enfant  If [ curStatus == 0 ]  Then  Echo « Je suis l’enfant »  Fi |

### Donnez un exemple détaillé commenté de démarrage d’un « Thread »

|  |
| --- |
| #include <pthread.h>  //Voici une création de base  //int pthread\_create(pthread\_t \*restrict tidp, const pthread\_attr\_t \*restrict attr, //void \*(\*start\_rtn)(void), void \*restrict arg);  Pthread processus;  Pthread\_create(&processus, NULL, close /\*Simple action, mais pourrait être une fonction\*/, /\*Les paramètres de la fonction si nécessaires.\*/) |

### De quelle façon doit-on compiler un programme utilisant des « Thread » ?

|  |
| --- |
| gcc MyProgram.c -o MyProgram -lpthread |

### Est-il possible de passer des paramètres ou variables à un « Thread » ? Si « Oui » comment ? Si « Non » pourquoi ?

|  |
| --- |
| Il faut les passer dans le thread\_create() |

## 2- Compilez le programme suivant et par la suite commentez à chaque ligne débutant avec un symbole de commentaire, de façon à bien expliquer le fonctionnement du programme.

//Ajout des librairies

#include <pthread.h>

#include <stdio.h>

/\*Fonction dont le rôle est d'incrémenter l'adresse d'un pointer vers un entier\*/

void \*inc\_x(void \*x\_void\_ptr)

{

/\*Pointeur d'entier dans lequel on assigne l'adresse d'un autre pointeur de méthode void\*/

int \*x\_ptr = (int \*)x\_void\_ptr;

/\*Tant que la valeur incrémentée est inférieur à 100, incrémenter encore\*/

while(++(\*x\_ptr) < 100);

/\*Afficher message à l'utilisateur comme quoi l'incrémentation s'est bien déroulée et qu'elle \*/

printf("x increment finished\n");

/\*Retour null\*/

return NULL;

}

/\*voici la fonction main dont la valeur de retour équivaut à un chiffre différent de zéro si un problème survient et de zéro si tout va bien\*/

int main()

{

/\*Déclaration d'entiers\*/

int x = 0, y = 0;

/\*Afficher à l'écran la valeur de x et la valeur de y\*/

printf("x: %d, y: %d\n", x, y);

/\*Déclaration du processus\*/

pthread\_t inc\_x\_thread;

/\*Si le processus n'a pas été créé\*/

if (pthread\_create/\*La commande de création du processus\*/(&inc\_x\_thread/\*Le processus à créer\*/, NULL/\*Valeur de retour attendue\*/, inc\_x/\*La fonction\*/, &x/\*Paramètre entré dans la fonction\*/))

{

/\*Afficher message d'erreur\*/

fprintf(stderr, "Error creating thread\n");

/\*Erreur, retourner chiffre différent de zéro\*/

return 1;

}

/\*Tant que la valeur incrémentée est inférieur à 100, incrémenter encore\*/

while (++y < 100);

printf("y increment finished\n");

/\*Si la suspension échoue\*/

if (pthread\_join(inc\_x\_thread, NULL)) /\*La fonction join arrête le programme jusqu'à ce que le processus cible soit terminé\*/

{

/\*Imprimer message d'erreur\*/

fprintf(stderr, "Error joining thread\n");

/\*Erreur, retourner chiffre différent de zéro\*/

return 2;

}

/\*Afficher à l'écran la valeur de x et la valeur de y\*/

printf("x: %d, y: %d\n", x, y);

/\*Tout s'est bien passé, retourner 0\*/

return 0;

}

## 3- Exemple de programme multitâche :

### Vous devez créer un programme C qui :

* Demande à l’utilisateur combien de « thread » il désire créer, un minimum de 2 et maximum de 10.
* Valider le nombre entré.
* À partir du nombre entré, créer autant de « thread » que demandé et chaque « thread » :
  + Affichera son numéro, de 2 à 10.
  + Effectuera une attente correspondante à un nombre de secondes 2 fois plus grand que son numéro.
  + Affichera un message de fin d’attente
  + Terminera
* Lorsque toutes les « thread » sont terminées, un message de fin de programme s’affiche.