Informatique embarqué

TP n°5 - MION Giovanni

Temps estimé: 4 heures

Temps effectif: 4 heures

I)Compilation du code

Pas de code

II)Choix de l'implémentation

Pas d'implémentation

III)Problèmes rencontrés

Voici une liste des problèmes rencontrés:

- Questions sur la gestion des signaux
- Question dépendant de l'architecture 32/64 bits

IV)Question du TP

Q1 - On ne s’arrête pas si argc = 1 et on demande argv[1] se qui sous-entend que argc >1.

Si il n'y a pas d'argument le programme affiche le message d'erreur puis plante.

Correction: mettre un else qui encadre le reste de code.

Q2 - Inversion des paramètres par rapport a l'affichage

De plus, le pointeur chaîne a été déplacé avec ++ jusqu'à la fin de la chaîne de caractères donc lors du printf nous avons un accès mémoire erroné.

Correction : faire une copie de la chaîne avant de la parcourir.

Q3 - Les membre de struct timeval sont des long int et sont caster en long dans le code.

La taille des long n'est pas la même selon que l'on se trouve sur 32 ou 64 bits.

Il se peut donc que la valeur du long int dépasse la valeur max d'un long et induise un problème sur une machine 32bits alors que non sur une machine 64bits.

Correction : utiliser des long int partout.

Q4 – La condition du if est « res = 0 » ce qui n'est pas un teste mais une affectation.

Correction : res == 0

Q5 – Il n'y a pas de vérification du retour de malloc avant d'utiliser strcpy. Si malloc renvoi null on a un core dump.

Correction : quittez le programme si le malloc renvoi null avec un message d'erreur.

Q6 - On crée plus que 20 processus car après le fork il n'y pas de différentiation des code du père et du fils ce qui fait que le fils aussi refait des appelle à fork.

Correction : utiliser la valeur de retour du fork pour différentier le père et le fils et évité que le fils fasse des appelle à fork.

Q7 - Chaque thread crée recommence la fonction de création de thread (threadception).

Correction : utilisé une autre fonction que my_funk dans pthread_create.

Q8 – Je ne sais pas :(

Q9 - Il se peut que après l'appelle du fork, le processus actif soit le fils dans ce cas il quitte et génère le signal SIGCHLD qui appelle le signal handler du père puis relance celui-ci juste avant l'appelle à pause() du père se qui le fait attendre un signal qui est déjà arrivé.

Correction : il faut s'assurer que le père rentre en pause avant que le fils ne quitte avec des moyens de synchronisation comme yield ou sleep pour endormir le fils et donné la main au père en premier.

Q10 – Il n'y a pas de teste de retour de malloc avant d'accéder au champs de la structures ce qui peut provoquer un core dump si malloc renvoi null.

Correction : quitter le programme avec un message d'erreur si malloc renvoi null

Q11 – Il y a un appelle à free(pt) dans colimacon. On renvoi donc un pointeur sur une zone de mémoire désallouer se qui créera un core dump si on l'utilise.

Correction : supprimer l'appelle a free.

Q12 - La fonction colimacon peut renvoyer null si malloc échoue et dans la fonction main on utilise le tableau retourner sans vérifier si il est non null.

Correction : vérifier que le tableau renvoyer et non null et dans le cas échéants ne pas l'utilisé (faire ko++ directement ou quitté avec un message d'erreur).

V)Commentaire d'un autre étudiant

Ce TP étant des réponses à des questions, il n'a pas été relut