Système d'exploitation

BARRERE Louis-Gabriel CATALDI Alexis

September 2019

Contents

1 Bilan

Un fichier Réponse.txt (dans code/) contient l'intégralité de nos réponses aux questions du TP, ainsi que les détails des fichiers modifiés pour y répondre si nécessaire.

Nous avons géré la partie multithreading utilisateur en créant/modifiant userthread.h, userthread.cc, syscall.h, exception.cc, Makefile.common et start.S. La création des threads a été définie pour des appels système comme pour le précédent TP et est utilisé par do_threadCreate() et do_threadExit(). Par ailleurs, des DEBUG() ont été utilisés fréquemment en début de fonction ou de case (ou à des endroits où l'on a fait nos implémentations) sauf dans le répertoire test car ça n'était pas nécessaire.

Les appels système ont été définis avec les nombres-codes 15 et 16 et leurs prototypes sont déclarés dans syscall.h. Les appels systèmes ThreadCreate() et ThreadExit() ont été définis en assembleur et leurs exceptions sont traitées dans le switch de exception.cc. Tous ces appels fonctionnent correctement.

Afin de tester nos implémentations de la partie 1, nous avons mis en place notre fichier de test makethreads.c dans lequel nous avons un main qui bloque sur un while 1 afin de permettre au thread créer de s'éxécuter avant la fin du programme principale. De ce fait nous devons ensuite manuellement faire "crasher" le programme principal via l'utilisation de "Control+C".

Les sémaphores de la partie 2 du projet fonctionnent correctement d'après plusieurs tests menés, ils ont posés cependant des problèmes ailleurs... Ceci est détaillé dans les points délicats et limitations plus bas. Pour la suite de la partie 2, tous nos tests ont été fait dans le makethreads.c déjà créé. Le reste (partie 2.4 et bonus) n'est pas encore implémenté.

2 Points délicats

Lors de l'implémentation de la question 1 de la partie 2, nous avons eu un problème avec PutString().

Nous avons donc effectué différents petits tests rapides pour voir si l'erreur se comportait toujours de la même manière et ainsi déterminer d'où l'erreur pouvait provenir, nos tests consistaient principalement à demander l'écriture de différentes chaînes de charactères par le biais de PutString(), les résultats obtenus sont les suivant:

- "autre" et "test" = "aesttest"
- "main" et "thread" = "mhreadthread"
- "a" et "t" = "at"

Bien que les sémaphores fonctionnent d'après les tests, les chaînes écrites par des threads différents suivaient toujours le même patern : écrire la première lettre de la chaîne passée au PutString() du thread main, puis toutes les lettre du PutString() du thread créé sauf la première et enfin réécrire cette dernière mais au complet cette fois. Sans les sémaphores, les caractères dans "aesttest" seraient tous écrits mais dans un ordre quelconque alors qu'avec les sémaphores le patern est toujours le même.

Au final, à cause de ce problème, il nous était impossible de savoir si nos fonctions GetChar() et PutChar() fonctionnaient correctement car le patern présenté ci-dessus ne change pas le résultat final obtenu. Heureusement le problème a été corrigé récemment et tout s'affiche correctement, le problème a consommé beaucoup de notre temps pour le résoudre.

3 Limitations

Les tests sur les appels Get n'ont pas été fait, puisque nous nous sommes principalement concentrés sur le problème énoncé dans la partie "Points délicats" du rapport.

4 Tests

Comme tous les tests de cette partie concernait des threads, nous avons décidé de faire tous les tests de la partie 2 dans un seul fichier pour vérifier le fonctionnement du projet (accessible dans test/) :

- makethreads.c : teste si les appels à PutChar() et putString() fonctionnent avec plusieurs threads et qu'ils "n'empiètent pas" les uns sur les autres.

Les lignes de commande pour lancer les tests se trouvent dans les fichiers test directement en commentaire.