Rapport Système d'Exploitation

Multithreading

9 novembre

TISNÉ Romain - VIGUÉ Thomas

1. Bilan

1.1 Partie I

Afin de permettre à un utilisateur d'utiliser des threads dans un programme Nachos, on a essayé de comprendre dans un premier temps comment était implémenter les threads Nachos.

Un thread noyau est alloué avec une pointeur de pile et un pointeur de sommet de pile initialisé à NULL et des registres initialisés à 0. La pile d'un thread Nachos en tant que thread noyau se trouve dans la mémoire linux.

Pour permettre à un programme utilisateur d'utiliser des threads on a mis en placer deux appels système ThreadCreate et ThreadExit (même procédure que dans le premier devoir). La création d'un thread peut échouer s'il n'y a pas assez de place dans la pile de threads d'un processus, par défaut la taille allouée aux threads est de 1024 octets (la taille d'un thread est de 256 octets).

ThreadCreate prends deux paramètres, on récupère le premier dans le registre 4 de la machine et le deuxième dans le registre 5. Ces deux registres sont envoyé dans la fonction do_ThreadCreate (définie dans userprog/userthread.cc) qui initialise un nouveau thread. Ce thread est placé dans la file d'attente des threads avec la méthode Start qui prend en paramètre StartUserThread et tableau de pointeur (schmurtz) qui contient la un pointeur sur fonction que doit exécuter le thread et un pointeur sur les arguments de cette fonction.

StartUserThread met tous les registre MIPS à 0 et initialise les registres PCReg, NextPCReg, StackReg, et met dans le registre 4 les arguments puis lance la machine. On met dans le registre de pile du thread (StackReg) l'adresse du haut de la nouvelle pile donnée par la méthode AllocateUserStack (définie dans userprog/addrspace.cc)

En ce qui concerne l'appel système ThreadExit on implémente la fonction do_ThreadExit (définie dans userprog/userthread.cc) qui termine le thread courant. On doit en plus libérer son espace d'adressage space.

Le fonctionnement de notre code est testé dans un petit programme de test (dans test/makethreads.c). Pour l'instant on test avec un seul thread et avec un simple PutChar exécuter par le thread créer. Le thread créer et le thread principale (le main) sont terminer par l'appel à ThreadExit.

1.2 Partie II

Dans cette partie on veut pouvoir lancer plusieurs threads par processus, dans la limite de quatre (la taille de la pile est de 1024 octets). A ce niveau du projet lorsque l'on essaye de créer plusieurs threads dans notre programme de test on obtient un message d'erreur Assertion failed.

Pour régler le problème on doit utiliser un verrou pour protéger les fonctions noyau. Pour ce faire on a implémenter les locks en s'inspirant de l'implémentation des sémaphores. L'idée c'est qu'un lock est un mutex (un sémaphore initialisé à 1) qui est soit occupé soit libre. Donc à la place d'une value (dans l'implémentation du sémaphore) on a un booléen busy initialisé à false.

On a donc placé les fonction SyncPutChar et SyncGetChar de la classe SyncConsole dans une section critique. Pour ce faire on utilise deux verrous différents mutexRead mutexWrite. On n'a pas trouvé nécessaire de mettre des verrous sur les méthodes SyncPutString et SyncGetString vue que ces méthodes appellent SyncPutChar et SyncGetChar.

On a ajouté un compteur de thread dans do_ThreadCreat et do_ThreadExit afin d'arrête la machine si le thread dernier est terminé.

2. Points délicats

La création des threads utilisateur nous a posé pas mal de souci, notre manque de compréhension de la totalité du sujet nous a empêché de finir le projet.

3. Limitations

On n'a pas réussi à faire en sorte que l'on puisse lancer plusieurs threads sans que l'on obtienne des comportements non désirer.

De même on n'a pas utilisé de bipmap pour mieux allouer les piles.

On n'a pas pu faire les parties bonus. Donc on est obligé de terminer les threads par ThreadExit, et on n'a pas fait les appels systèmes pour que les sémaphores soient utilisables par utilisateur.

4. Tests

Notre unique fichier de test est test/makethreads.c

On peut l’exécuter avec la commande (à partir du dossier code)

```
./userprog/nachos -rs 1234 -x test/makethreads
```