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 asynchrone en modifiant progtest.cc : le "Good Bye !" s'affiche si q est lu, puis pour afficher les caractères entre chevrons au lieu de seulement afficher x (un caractère quelconque), PutChar est appelé si le retour à la ligne n'est pas lu. Pour que ceci fonctionne aussi avec des fichiers (dans notre cas, input.txt et output.txt), nous avons ajouté synchconsole = new SynchConsole (NULL ,NULL); dans le main.cc afin de spécifier que des paramètres d'entré peuvent être donnés ou non.

En ce qui concerne la partie synchrone, les fonctions PutChar et GetChar ont été implémentées dans synchonsole.cc à l'image du .h. SynchconsoleTest a été implémenté comme suit : on prend une entrée et une sortie en paramètre pour les donner à Synchconsole, puis tant que la fin de fichier n'est pas détectée on affiche un caractère entre chevrons.

Les appels système ont été définis avec des nombres-codes de 11 à 14 et leurs prototypes sont déclarés dans syscall.h. Les appels systèmes PutChar(), PutString(), GetChar() et GetString() ont été définis en assembleur et leurs exceptions sont traitées dans le switch de exception.cc. Tous ces appels fonctionnent correctement.

On a choisi d'implémenter copyStringFromMachine() et copyStringToMachine() dans system.h pour qu'elle soit accessible depuis console.h et synchconsole.h, de plus on devait avoir accès à machine (depuis machine.h).

Exit() a été géré afin d'éviter d'avoir une erreur si on ne fait pas appel à Halt() et le code de retour a aussi été géré dans 2 fichiers créés à l'occasion : exit.c et return.c (et l'assembleur + les exceptions).

Au final, les appels PutChar(), GetChar et PutString ont été implémentés, testés et fonctionnent.

2 Points délicats

int copyStringFromMachine(int from, char *to, unsigned size); comporte des paramètres que l'on a eu du mal à comprendre : "from" et "to" ont des noms suffisamment représentatifs pour qu'on les comprenne, mais les types associés sont étranges.

Ce n'est qu'après de nombreuses tentatives que nous avons compris que "from" est directement une adresse (on s'attendait surtout à avoir un pointeur pour stocker l'adresse plutôt que d'avoir directement un int). C'est voulu pour ne pas toucher à la valeur associée à cette adresse selon nous.

void GetString(char *s, int n); nous a posé beaucoup de difficultés à implémenter. Nous n'arrivions pas à voir pourquoi notre implémentation ne fonctionnait pas comme on le souhaitait. Elle est cependant finie à temps, mais le cas de concurrence n'a pas été géré.

3 Limitations

Les tests sur les appels Put sont limités : ils traitent de cas particuliers contrairement aux Get qui récupèrent les chaînes saisient dynamiquement pendant les tests.

Les appels concurrents ne sont pas gérés, ce qui implique que si plusieurs threads font appel à GetString() il est possible que le buffer ne contienne pas le résultat attendu.

Les accents ne sont pas gérés (et d'autres caractères spéciaux) : certains caractères donnent un caractère imprévu comme '?' au lieu de 'é' par exemple.

4 Tests

A chaque implémentation nous avons décidé de créer un fichier spécifique pour les tester (accessible dans test/) :

- getchar.c : teste l'implémentation de GetChar
- putchar.c : teste l'implémentation de PutChar
- putstring.c : teste l'implémentation de PutString
- getString.c : teste l'implémentation de GetString
- halt.c : teste l'implémentation de Halt
- exit.c : teste l'implémentation de Exit
- return.c : teste le code renvoyé par return

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