Rapport de projet ASR2

GEDEON Elie - PY-CIRCAN Etienne

Howto

Pour compiler, taper make dans le répertoire code.

Dans le répertoire code, lancer ./userprog/nachos -x ./test/putchar pour le test générique. Son fonctionnement est décrit dans ./test/putchar.c.
Tous les autres tests proposés par le sujet sont fonctionnels, sauf ceux qui impliquent un conflit console asynchrone/synchrone.

Remarques, problèmes et solutions

Partie I

On s'attend à l'affichage de "abcd" à l'écran.

Problèmes :

Pas de problèmes rencontrés

Choix :

Sans objet

Partie II

Si on essaie de lire un caractère alors que l'on n'a pas été prévenu de sa présence, il peut ne pas y avoir de caractère à lire, auquel cas ce qu'on lit n'a pas de sens, et pourrait être une donnée d'un autre processus, qui n'est pas censée être divulgée, ou encore une donnée précédemment écrite ou lue.

Problèmes :

Il fut difficile de faire fonctionner nachos avec in et out. En fait, seuls certains exécutables nachos fonctionnent. Quelles sont leur différences ? Nous ne savons pas. En dernière date, cela ne marche pas avec ./userprog/nachos.

Choix :

-Le type de "ch" a été changé en int pour pouvoir renvoyer un EOF.
-Pour des raisons esthétiques, le caractère de retour à la ligne n'a pas été entouré de < >.

Partie III :

Il s'agissait uniquement de masquer la gestion des sémaphores : Un sémaphore signale l'arrivée d'un caractère, qui est remis à zéro à la lecture.
Toute cette partie consista en un grand "copier-coller", réutilisant les fonctions précédemment définies.

Problèmes :

Pas de problèmes rencontrés

Choix :

les < > ont fini par être enlevés, ils rendaient la lecture des tests assez pénibles.

Partie IV :

Problèmes :

Bien comprendre où initialiser synchconsole *, ainsi que l'utilisation des fonctions dans start.S.
Une fois le mécanisme d'exception compris, peu de problèmes se sont posés, dans la mesure où la partie III marche.

Choix :

Tous les accès en écriture (respectivement en lecture) ont été protégé par des mécanismes de lock unlock déclarés dans synchconsole et utilisé dans exception.cc

Partie V :

Synchputstring consiste en une simple boucle while, qui ne modifie pas son argument.

Problèmes :

Le respect des "const" dans les arguments des fonctions synchrones fut pénible.
On ne peut utiliser directement to car il pointe dans une zone mémoire MIPS. Y accéder directement reviendrait à courcircuiter le simulateur, ce qui n'est pas désirable.
Si on allouait à la demande le buffer, il y aurait un risque de plantage du noyau à cause d'une allocation trop grosse : un bug dans un programme (ou une malveillance) pourrait conduire à une chaine sans caractère nul, de longueur « infinie »

Choix :

Le buffer a une taille maximale comme suggéré dans le TP. Il est initialisé dans system.cc à la création du
copyStringFromMachine écrit au plus MAX_STRING_SIZE caractères. Cependant si il ne rencontre pas de caractère '\0', il le rajoutera après, ceci pour différencier une copie réussie (toute la chaîne a été copiée) d'une copie inachevée.
L'appel système PutString a été modifié dans exception.cc pour redemander une impression tant que la chaîne entière n'a pas été imprimée.
copyStringFromMachine a été placé dans SynchConsole.cc vu qu'il y est directement lié.

Partie VI :

Problèmes :

Lors de l'arrêt du programme, une erreur signale qu'une exception n'a pas été rattrapée. L'exception SC_Exit est lancée à la fin du programme. Il faut donc la rattraper. Pour récupérer la valeur de retour, il faut mettre la valeur de r2 (valeur de retour du main) dans r4 (paramètre de l'appel système).

Choix :

Une fonction Exit a été ajoutée dans interrupt.cc. Exit affiche grâce à printf, car on est dans le domaine du debuggage et de la simulation et non de nachos.

Partie VII :

En cas de fin de fichier, GetChar renvoie -1.
GetString alloue dans exception.cc un buffer de la bonne taille qu'elle désallouera après.

Problèmes :

Pas de problèmes rencontrés

Choix :

une fonction copyToString a été ajoutée dans synch_console.cc.
le buffer pour getint et putint est de 50 octets ce qui est largement suffisant.

Partie VIII :

Problèmes :

stdarg.h n'existe pas

vsprintf.c est compilé comme un exécutable.

La taille du buffer n'est pas connue d'avance.

Choix :

inclusion directe de stdarg.h

modification manuelle du Makefile

vprintf écrit directement caractère par caractère en passant par un buffer temporaire.