Université de Bordeaux
Département Master

NACHOS:Entrées/Sorties

ANABA Henri Frank
CONSTANS Olivier
TOUATI Amira

1ère Année de Master Informatique
2014-2015

Table des matières

Chapitre 1

Implémentation

1.1 Entrées-sorties

La première version d'Entrées-sorties était asynchrone. L'utilisateur devait se préoccuper de la synchronisation de ces lectures/écritures en utilisant des **"Semaphores"** . Lire sans être averti qu'il y ait quelque chose à lire, ne garantit pas le bon résultat de la lecture.De même écrire sans être averti que l'écriture précédente est terminée peut provoquer un écrasement de caractères.

Nous avons donc implémenté une couche entrées-sorties synchrone *SynchConsole* au-dessus de la couche *Console* qui encapsule le mécanisme des sémaphores. Ainsi l'utilisateur a à sa disposition deux fonctions qui font la synchronisation pour lui.

On a donc complété le fichier *Makefile.common* pour rajouter *SynchConsole* et modifier le fichier *threads/main.cc* pour ajouter une option -sc de test de la *SynchConsole* .

1.2 Appel système PutChar

Pour le bon fonctionnement du programme utilisateur Nachos *PutChar* on a du mettre en place un appel système dans le fichier *userprog/exception.cc* en utilisant la fonction *SynchPutChar* définie dans la couche *SynchConsole* .

Ensuite on a été amené à ajouter le numéro (11) de l'appel système et la déclaration de la fonction dans le fichier *userprog/syscall.h* puis à rajouter

la définition en assembleur de PutChar dans le fichier *test/start.S* .

Mais pour que ça marche on a du ajouter une définition globale de *SynchConsole* dans l'initialisation du système dans le fichier *threads/system.cc* .

1.3 Des caractères aux chaînes

Pour le moment notre système d'exploitation ne peut écrire qu'un seul caractère à la fois, on s'est donc occupé de compléter la méthode *SynchPutString* de la classe *SynchConsole* puis d'implémenter la procédure *copyStringFromMachine* qui copie une chaîne de caractère du mode MIPS vers le mode Linux. Nous avons mis cette procédure dans le fichier *userprog/exception.cc* car elle fournit au noyau une copie de la chaîne utilisateur. Cette copie sera réservée au système.

on a ajouté l'appel système *PutString* qui utilise les procédures *copyStringFromMachine* et *SynchPutString* , pour cela on a utilisé un buffer local de taille *MAX_STRING_SIZE* en la déclarant dans le fichier *threads/system.h*

Il ne serait pas raisonnable d'allouer un buffer de la même taille que la chaîne MIPS ; car si l'utilisateur demande une chaîne de caractère très grande, ça va allouer trop de mémoire et pour optimiser cela on a préféré utilisé un buffer interne. Si la taille du buffer interne est inférieure à celle de la chaîne, nous lisons la chaîne par morceau.

1.4 Modification de fin de fonction

Si un programme utilisateur oubliait d'utiliser l'appel système *Halt*, cela provoquait une erreur. Pour corriger ce problème, nous avons rajouté le comportement de *Halt* par default. Si un utilisateur oublie d'utiliser *Halt*, le système utilise l'appel système *Exit*. Nous avons donc donné le comportement de *Halt* à *Exit*.

De plus, notre système d'exploitation ne prenez pas en compte la valeur de retour. Cette erreur était dû a un problème dans le code assembleur, lors de l'appel système*Halt* le système ne s'arrête pas directement, il appelle une

dernière fois *Exit*. Or la valeur de retour était stockée dans le registre 2 et l'appel de *Exit* écrasait la valeur du registre 2. Ainsi pour régler le problème,nous avons copié la valeur de retour dans le registre 4 avant de faire l'appel de *Exit*.

1.5 Fonctions de lecture

A ce moment notre système d'exploitation ne sait qu'écrire , pour lui permettre de lire on a complété l'appel système *GetChar* dans le fichier *userprog/exception.cc* en affectant le résultat de la fonction *SynchGetChar* dans le registre 2.

Ensuite on a complété la méthode *SynchGetString* de la classe *SynchConsole* pour permettre la lecture d'une chaîne de caractère. Puis complété l'appel système *GetString* et il a fallu définir une fonction *copyStringToMachine* qui permet de copier la chaîne dans le registre 4.

1.6 Gestion des entiers

Afin de permettre au système d'exploitation l'écriture externe décimale on a ajouté l'appel système *PutInt* en utilisant la fonction *snprintf* qui permet de convertir un entier en chaîne de caractère pour la donner en paramètre à la fonction *SynchPutString*. De même pour *GetInt*,nous avons utilisé la fonction *SynchGetString* et nous avons converti la chaîne récupérée en entier grâce à la fonction *sscanf*.

Chapitre 2

Points délicats et limitation

2.1 Points délicats

Ce qui a été plus complexe à réaliser dans le projet était la partie concernant les appels système *PutString* et *GetString* . On a dû utiliser un buffer de taille constante pour l'écriture et la lecture, quelque soit la longueur de notre chaîne.
Pour résoudre ce problème on a utilisé une boucle qui compare le taille du buffer à la taille de la chaîne restante à lire. Si la taille du buffer est inférieure on lit par segment de ce dernier, sinon la taille de la chaîne restante.

Nous avons eu un autre problème concernant le *GetString* , nous avions mal compris le comportement qu'il devait avoir dans le cas où la chaîne de caractères donnée est plus grande que le nombre de caractère à lire. Pour éviter que les caractères non lus se retrouve sur la console, nous avions décidé de les lire sans les mémoriser. Hors le comportement attendu de *GetString* ne se préoccupe pas des caractères excédants. Nous avons donc décidé de ne plus les lire.

2.2 Limitation

Pour pouvoir afficher simultanément des chaîne de caractères et des entiers, l'utilisateur doit utiliser consécutivement les fonctions *PutString* et *PutInt* . Pour régler ce problème nous aurions pu implémenter la fonction *printf* .

Chapitre 3

Tests