Martial DUVERNEIX
Teiki PEPIN

Rapport
Systèmes d'exploitation
Projet 1 : Entrées/Sorties

I – Bilan :

Dans le cadre de la mise en place d'un mécanisme d'entrée-sortie sous Nachos, nous avons commencé par l'étude des fichiers sources de la console fournis. Nous avons procédé à des modifications basiques à ces fichiers afin de changer le comportement de la console. Ensuite, nous avons créé une console synchrone dans le but est d'encapsuler un mécanisme d'entrées-sorties synchrones à base de sémaphores. Nous avons implémenté dans un fichier C les fonctions *SynchConsole*, *~SynchConsole*, *SynchPutChar*, *SynchGetChar* et ajouté ce fichier au *Makefile* afin que notre nouvelle console soit compilée avec le reste des fichiers de Nachos. Nous avons modifié les fichiers *main.cc* et *progtest.cc* afin de permettre à Nachos de lancer notre console synchrone.

Par la suite, nous avons rajouté un appel système dans Nachos permettant de lancer notre fonction *SynchPutChar* en éditant le fichier *syscall.h*. Nous avons également rajouté la définition en assembleur de notre appel système *PutChar*. Nous avons permis à Nachos de gérer notre appel système en tant qu'exception non critique en ajoutant un appel à notre console dans le fichier *exception.cc* et en la définissant lors de l'initialisation du système via *system.cc*. Cela nous permet de lancer des programmes de test fonctionnels qui font plusieurs appels à notre appel système.

Nous avons ensuite continué avec la suite logique à ces modifications en rajoutant à notre *SynchConsole* une fonction *SynchPutString* permettant d'afficher une chaîne de caractères en entrée en se basant sur plusieurs *SynchPutChar* successifs. De même, un appel système correspondant est ajouté. Celui-ci nécessite l'utilisation d'une autre fonction (*copyStringFromMachine)* que nous avons créé permettant la traduction caractère par caractère du monde utilisateur MIPS vers le monde noyau.

De façon similaire, nous avons ensuite implémenté les fonctions *SynchGetChar* et *SynchGetString* et les appels systèmes *GetChar* et *GetString*. Ces fonctions permettent, comme leur nom l'indique, de recupérer un caractère et une chaîne de caractère respectivement. *GetString* fait appel à une nouvelle fonction *copyStringToMachine* qui fait la même traduction que la fonction *copyStringFromMachine* mais en sens inverse.

Enfin, nous avons retiré les appels à *Halt( )* de nos programmes de test. L'appel système *Exit( )* est désormais appelé par Nachos après l'exécution de nos programmes utilisateurs mais celui-ci n'est pas une exception reconnue. Nous avons donc rajouté une gestion du cas *SC_Exit* à notre fichier *exception.cc* qui fait simplement un appel à *Halt( )* afin de se débarrasser des appels manuels à *Halt* de nos fichiers de test et des erreurs conséquentes. De même, si l'on souhaite récupérer la valeur de retour des *main* de nos programmes, nous pouvons le faire dans le fichier *shell.c* au niveau du *Join* qui attend la fin du processus donné en argument avant de renvoyer sa valeur de retour. Le main du fichier *shell* est initialement appelé dans la routine *__start* de *start.S*.

II – Points délicats :

Parmi le travail effectué, plusieurs sections ont posé des difficultés. Initialement, la mise au point de *SynchConsole* a causé des soucis dans la manière dont elle devait implémenter la console originale de Nachos et émuler les fonctions d'entrées-sorties Unix (*putchar*, *getchar* et *fgets*). Après plusieurs tentatives, une répartition fonctionnelle du code entre la console synchrone et les fonctions de la consoles standard a été trouvée, permettant ainsi de faire fonctionner la console avec nos sémaphores.

Un autre point délicat de ce projet était la lecture et la modification du fichier *start.S*. Ce-dernier étant écrit en langage assembleur. Le langage nous étant majoritairement inconnu à tous les deux, le travail de compréhension du fichier a été accompagné de beaucoup de recherche et de documentation en ligne. Cela nous a permis de comprendre les modifications que nous y avons apporté avec *PutChar*, *PutString*, *GetChar* et *GetString*, et surtout le déroulement des étapes à l'initialisation via *__start*.

L'écriture de la procédure *int copyStringFromMachine(int from, char *to, unsigned size)* s'est avérée particulièrement difficile. Celle-ci faisant la traduction des adresses utilisateurs virtuelles aux adresses physiques en utilisant la fonction *ReadMem*, en faisant des conversions de *int* à *char*, en se souciant de la taille de la chaîne, de la taille désignée par l'utilisateur, et de la taille maximum permis par l'implémentation. La combinaison des ces facteurs délicats nous a initialement bloqué jusqu'à que chacun de ces points ne soit parfaitement compris et qu'un code fonctionnel soit implémenté. On notera, entre autres, le fait que la fonction *ReadMem* a pour argument *int *value*, ce qui nous oblige a cast notre adresse de destination to dû à la différence de représentation en mémoire entre les *ints* et les *chars*. La fonction similaire *copyStringToMachine* n'a ainsi pas posé de problème, celle-ci réutilisant principalement les points clés précédemment mentionnés.

La dernière difficulté conséquente que nous avons rencontré au cours de ce projet concerne la suppression de l'appel à *Halt( )* de nos programmes de test. Cela provoquait l'affichage d'un message d'erreur ("*Unimplemented system call*") à la fin de l'exécution de nos programmes. Nous n'avions pas immédiatement remarqué que le message d'erreur était produit par *exception.cc* et que l'identifiant d'appel système fourni nous permettrait d'identifier que Nachos tentait d'appeler *Exit( )*. Ce n'est que bien plus tard que nous avons fait ce lien et avons résolu le problème par l'ajout du cas de *SC_Exit* dans *exception.cc*. Nous avons d'ailleurs remarqué par la suite que l'appel à *Exit( )* se faisait dans *start.S*.

III- Limitations :

Il y a certains points clés de notre implémentation qui imposent des limitations. En cas de fin de fichier lors de notre appel *GetChar*, le caractère EOF est pris par notre console et la saisie est interrompue, le programme continuant ensuite avec le reste de son exécution.

Les fonctions *copyStringFromMachine* et *copyStringToMachine* ont été créées dans une librairie séparée afin de permettre leur réutilisation par la suite (*userprog/copystring.h* et *userprog/copystring.cc*).

De plus, ces fonctions utilise une constante *MAX_STRING_SIZE* que nous avons définie dans *system.h* (actuellement un entier de valeur 100). Cette constante est aussi utilisée dans *exception.cc*. Cette double utilisation de la constante afin d'éviter un dépassement de tampon en mémoire est redondante mais nous permettrait d'utiliser les fonctions *copyString…* en dehors de *exception.cc* ou de remplacer ces fonctions dans ce même fichier sans perdre en robustesse dans notre implémentation.

Nos implémentations des appels système *PutString* et *GetString* font usage d'un buffer que l'on alloue au début et que l'on libère à la fin de chaque appel système afin d'éviter des fuites mémoire.

IV - Tests :

Nous avons mis en place un programme de test pour chaque appel système implémenté. Le programme se trouvant dans le fichier *putchar.c* affiche à l'écran plusieurs caractères successifs avant de se terminer. Le fichier *getchar.c* permet de récupérer un caractère saisi par l'utilisateur avant de l'afficher, permettant ainsi de vérifier que le caractère a été correctement enregistré. Le fichier *putstring.c* affiche plusieurs chaînes de caractères quelconques. La taille des chaînes affichées peut varier afin de tester l'efficacité de la limitation de taille implémentée comme décrit précédemment. Le fichier *getstring.c* nous permet de récupérer une chaîne de caractère saisie par l'utilisateur (la saisie se termine lors d'un retour à la ligne dans la console) avant de l'afficher à l'écran pour confirmer qu'il a été correctement enregistré. La taille de la chaîne saisie est ici aussi modifiable afin de permettre de tester la résistance aux dépassement de tampon.