Rapport Projet Nachos

Équipe H

Cédric GARCIA

Florian BARROIS

Hosseim NAHAL

Peio RIGAUX

Mohd Thaqif ABDULLAH HASIM

Table des Matières

Introduction

Le projet NachOS permet d'aborder quelques aspects d'un système d'exploitation par des étudiants. Il consiste en une émulation d'un système avec la partie noyau en C/C++ et une machine virtuelle MIPS qui permettra d'exécuter plusieurs processus. Il permet de comprendre le fonctionnement interne des systèmes d'exploitation, de gérer un grand logiciel, de travailler l'aspect gestion de projet en équipe et de programmer les principales fonctionnalités d'un système d'exploitation.

Vous découvrez tout au long de ce rapport les particularités de notre petit système d'exploitation.

Plusieurs aspects seront abordés : la présentation des fonctionnalités, des spécifications, des tests utilisateur, de l'implémentation, de l'organisation du travail.

Fonctionnalités principales

[à remplir]

Spécifications

Étape 2

void SynchPutChar(const char ch)

Écrit le caractère ch dans la sortie définie précédemment (à l'initialisation de SynchConsole())

char SynchGetChar()

Récupère et renvoie un caractère à partir de l'entrée définie précédemment.

void SynchGetString(char* s, int n)

Récupère n caractères de la chaîne de caractères via l'entrée définie précédemment et les stocke dans la chaîne s stockée en paramètre. La lecture s'arrête lorsqu'un '\n' ou EOF est rencontré.

'\0' est stocké après le dernier caractère dans le tampon s.

'\0' est stocké à l'indice 0 du tampon quand la fin du fichier est détectée alors qu'aucun caractère n'a été lu.

void SynchPutString(const char s[])

Parcourt la chaîne de caractères s et écrit caractère par caractère sur la sortie définie précédemment jusqu'à la rencontre du caractère '\0' ou jusqu'à ce que MAX_STRING_SIZE soit atteinte. Le comportement est indéfini dans le cas où s ne contient pas de '\0'.

void SynchGetInt(int n)

Lit un entier depuis l'entrée standard, si la valeur n'est pas dans $[-2^{31}; 2^{31}-1]$ elle sera forcée à une de ces valeurs (la limite est de 16 caractères pour les entiers, au-delà les caractères ne seront pas comptés). SI un utilisateur entre le signe '-' sans le faire suivre de chiffres, la valeur 0 est stockée.

void SynchPutInt(int * n)

Récupère la valeur de l'entrée définie précédemment puis l'écrit à l'adresse pointée par n. Le comportement est indéfini si la valeur n'est pas dans $[-2^{31}; 2^{31}-1]$.

Étape 3

int UserThreadCreate(void f(void *arg), void *arg)

Appel système appelant do_UserThreadCreate(). En cas d'échec, renvoie -1. Cette fonction échoue si le nombre maximum de threads a été atteint ou si il ne reste plus de place en mémoire lors de son appel (À VÉRIFIER). Initialise un thread utilisateur, l'ajoute à la liste de threads et exécute la fonction f. Au niveau des arguments, f pointe vers la fonction que devra exécuter le thread et arg correspond au paramètre passé à la fonction f. Lance StartUserThread() et renvoie l'identifiant du thread. La création d'un thread empêche la machine de s'arrêter jusqu'à ce que tous les thread aient fini de s'exécuter.

void UserThreadExit()

Appel système qui appelle do_UserThreadExit() afin de terminer un thread. Enlève le thread courant de la liste des thread et termine son exécution. Termine les threads si le main finit avant.

void UserThreadJoin(int Idthread)

Appel système qui appelle join_UserThread(int id). Il permet d'attendre la terminaison du thread portant l'identifiant Idthread à l'aide d'un sémaphore.

---

static void StartUserThread(int f)

Initialise les registres de la nouvelle machine MIPS et exécute la fonction f.

static int getIndexThreadById(int id)

Récupère l'index du thread ayant l'identifiant id dans la liste de threads.

Tests utilisateur

Test sur la console :

Pour l'instant, les tests fonctionnent à l'aide de l'entrée/sortie standard et les types de données pouvant être entrés sont testés manuellement (tentative de dépasser la taille que peut stocker un int pour getint.c par exemple).

testPutchar.c : Programme donné dans le sujet de l'étape 2 permettant de tester l'appel système PutChar et d'afficher une série de caractères. Il a été amélioré en testant plusieurs types d'entrées (entier, caractère, caractère non-ascii, caractère vide).

testPutString.c : Affiche la chaîne de caractères passé en paramètre de PutString() sur la sortie. Ce fichier teste plusieurs types de chaînes (vide, caractère non ascii, chaîne normale). Ce test permet de vérifier qu'une chaîne de caractères peut être affichée et de voir ce qui se passe si une chaîne de caractères dépasse la taille maxiumum.

testGetChar.c : Récupère un caractère sur l'entrée puis l'affiche, lui et les cinq caractères le suivant dans l'ordre alphabétique. Ce fichier teste ce fonctionnement pour un caractère et pour une chaîne de caractères.

testGetString.c : Demande le nom de l'utilisateur (une chaîne de caractères) puis l'affiche précédé de « Bonjour Monsieur ». Ce test permet de vérifier qu'un appel multiple à getChar() sur un tampon ne crée pas de problèmes.

testGetInt.c : Demande un entier à l'utilisateur, le borne afin de le faire rentrer dans un int, puis l'affiche (Ce test sert à la fois pour SynchGetInt() que pour SynchPutInt()).

makethreads.c : Programme donné dans le sujet de l'étape 3. Crée 6 threads et leur demande d'afficher chacun un caractère qui sera aussi affiché par le thread principal. Le thread principal affiche un caractère puis attend chaque thread alternativement. Ce test permet de vérifier que les threads se lancent, exécutent une fonction et sont bien attendues par le main en cas de join().

test3_threads.c : Création de deux threads affichant l'entier 44, le caractère '_' puis 5 ou 10 en fonction du thread 1 ou 2. Ce test permet de vérifier que l'on peut récupérer les informations concernant le thread et le non dépassement de pile.

test3_1threads.c : Création de threads qui vont afficher chacun une valeur différente t[i] d'un tableau t suivie d'un retour à la ligne. Le thread principal fait ensuite un join() pour tous les threads créés. Ce test simple permet de vérifier la validité du fonctionnement de UserThreadCreate(), sa valeur de retour, et UserThreadJoin().

Implémentation

Étape 2

char SynchGetChar() :

Fait appel à la fonction GetChar() de Console pour récupérer un caractère depuis l'entrée standard grâce un mécanisme d'interruption.

void SynchPutChar(const char ch) :

Fait appel à la fonction PutChar() de Console pour écrire un caractère ch sur la sortie standard déclenchant une interruption.

void SynchGetString() :

Effectue des appels à la fonction SynchGetChar() dans une boucle bornée par le nombre de caractères lus. Le caractère de fin de chaîne est concaténé à la chaîne obtenue.

void SynchPutString(const char s[]) :

Fait appel à la fonction SynchPutChar() dans une boucle pour afficher des caractères sur la sortie standard. La boucle s'arrête lorsque le caractère de fin de chaîne est détecté ou lorsque la nombre maximum de caractères est atteint.

void SynchGetInt(int *n) :

Lit une chaîne de caractères depuis l'entrée standard grâce à la fonction SynchGetChar(). Le premier caractère est testé pour pouvoir considérer les nombres négatifs. Ensuite tant qu'un caractère dont le code ASCII correspond à un chiffre est détecté, ce chiffre est ajouté à la chaîne. La chaîne est terminée par le caractère de fin de chaîne et est stockée dans l’entier pointé par n.

void SynchPutInt(const int n) :

Convertit l'entier n au format chaîne de caractères et l'affiche sur la sortie standard en appelant la fonction SynchPutString().

Étape 3

Organisation du travail en équipe

Répartition des tâches

Cédric : Conception et implémentation/rédaction des tests pour les parties 2,3 et 4.

Florian : Conception et implémentation pour les parties 3 et 4.

Hosseim : Conception et implémentation des parties 3 et 5.

Thaqif : Implémentation de la partie 2 et 6 et rédaction des tests.

Peio : Compte rendu et exécution des tests.

Les tâches assignées à chacun montrent l'organisation générale du groupe. Cependant, toute l'équipe se tient au courant de ce que chacun fait et peut ponctuellement travailler sur la même tâche (par exemple lors de la rédaction du premier compte rendu et de la compréhension du fonctionnement des piles de thread NachOS).

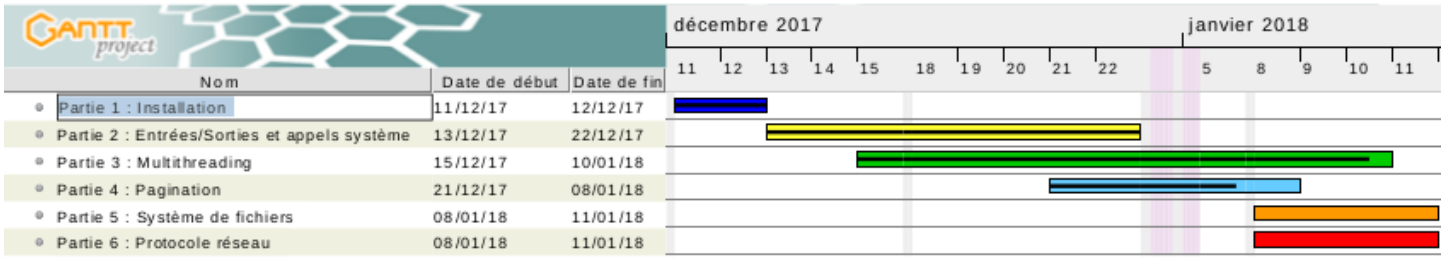

Diagramme de Gantt

Retour global sur le projet

[à remplir à la fin]