Équipe H

Rapport Projet Nachos

Cédric GARCIA Florian BARROIS Hosseim NAHAL Peio RIGAUX Mohd Thaqif ABD

11 janvier 2018

Table des matières

0.1 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.

Première partie

Fonctionnalités principales

[à remplir]

0.2 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 do_UserThreadCreate(int f, int arg) 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. Renvoie l'identifiant du thread. int do_UserThreadExit() Enlève le thread courant de la liste des thread et termine son exécution. static void StartUserThread(int f) Initialise les registres 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.

Deuxième partie

Tests utilisateur

0.3 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).

putchar.c : Programme donné dans le sujet de l'étape 2 permettant de tester l'appel système PutChar et d'afficher une série de caractères. putstring.c : Affiche la chaîne de caractères passé en paramètre de PutString sur la sortie. En cas de chaîne ayant une taille plus grande que MAX_STRING_SIZE, PutString n'envoie que les MAX_STRING_SIZE premiers caractères. getchar.c : Récupère un caractère sur l'entrée puis l'affiche, lui et les deux caractères le suivant dans l'ordre alphabétique. getstring.c : Demande le nom de l'utilisateur (une chaîne de caractères) puis l'affiche précédé de « Bonjour Monsieur ». getint.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()).

Troisième partie

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

Quatrième partie

Organisation du travail en équipe

Répartition des tâches

Cédric : Conception et implémentation, rédaction des tests. Florian : Conception et implémentation. Hosseim : Conception et implémentation. Thaqif : Implémentation, 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).

Cinquième partie

Retour global sur le projet

[à remplir à la fin]