Université de Bordeaux

Systèmes d'Exploitation rendu 3 : Pagination dans NACHOS

BECHER Iheb
YANG Yi

18 décembre 2021

Table des matières

Chapitre 1

Bilan

Lors de ce TP nous implémentons dans Nachos un modèle de processus qui gère la mémoire en pagination .Dans la partie I. Adressage virtuel par une table de pages, on a implémenté une mécanisme de **la table des pages** de l'utilisateur en utilisant la fonction Translate

Dans la partie II.Exe´cuter plusieurs programmes en meˆme temps,la fonctio 'int ForkExec(const char * s)' permet de lancer un nouveau processus qui exe´cute ce programme dedans.

Chapitre 2

Points délicats

2.1 ReadAtVirtual

Après analyse de fichier addrspace.cc , on remarque l'appel de la fonction readAt , qui manipule la mémoire physique . D'où l'intérêt de la fonction ReadAtVirtual . Tout d'abord on fait appel à la fonction readAt en passant un buffeur dans les paramètres . Ensuite on sauve la table des pages courante pour pouvoir la restaurer . En utilisant la fonction writeMem en copie le buffeur obtenu sur nos pages virtuelles . Il nous reste ensuite que de restaurer la table des pages . On avait eu des problème dans cette fonction en misant en palace un tampon temporaire dans l'espace noyau.

2.2 PageProvider

Dans la partie de la création de la classe PageProvider qui s'appuie sur la classe BitMap, on trouve la choix du nombre des pages physique était confus pour nous, on a essayé de la calculer nous même à l'aide des variable par exemple : la taille du page , Parce que au début on a négligé la relation entre le nombre des pages physique et le nombre de maximum thread donc la valeur MAXTHREADNUMBER qui est déjà défini dans system.h.

2.3 Stack Allocation

Dans cette partie, on a passé pas mal temps pour comprendre la changement du mémoire dont le stack en fonction d'effectuer plusieurs programme en même temps.

2.4 Appel System ForkExec

Comme d'habitude on met en place notre appel système ForkExec(char * ch)
Les fichiers de cet appel sont forkexec.cc et forkexec.h . C'était délicats de terminer le programme principale après les fork , et gérer les fuites mémoire .

2.5 Bonus

C'était délicats de faire tout les Bonus . On n'a pas pu tous les faire . Pour le bonus II.5. Quand on termine un processus de père en appelant Exit(-1) , tous ses fils exit aussi, mais ce partie nous

pose des warnning.

Chapitre 3

Limitations et Conclusion

On n'a pas implémente tout les bonus on espère que les fonctionnalités fournis et la qualité de code à la hauteur de vos attentes .

Plusieurs concepts sont abordés , et ça nous demandais beaucoup de réflexe et de de recherche . C'est pour ça le projet et projet était très intéressent et nous a permis de solidifier nos compétences et la programmation système et système d'exploitation .

Chapitre 4

Test

Dans /userprog lancer nachos par ./nachos -rs 1234 -x ../test/forkexec