Rapport du Devoir 3 : Pagination dans NACHOS
De Florian Claisse et Maxime Zielinger

BILAN

Questions du devoir

Partie 1

I.2.

Curieusement (?), on lui fait écrire directement en mémoire physique MIPS. À quoi voit-on cela ?

Il est visible que nous écrivons directement en mémoire physique car, grâce à la fonction ReadAt appelée dans AddrSpace, nous accédons directement à l'adresse dans la mémoire physique en passant par l'adresse de la mémoire virtuelle.

machine->mainMemory[noffH.code.virtualAddr] explicite ce passage du virtuel au physique.

I.5.

Pourquoi faut-il un seul objet de cette classe PageProvider ?

PageProvider permet l'encapsulation des pages physiques globales à tout nachos, par conséquent, si chaque processus possédait une version différente de PageProvider, ils utiliseraient toutes les pages comme s'ils étaient les seuls à y avoir accès.

I.6.

Quel type d'erreur peut survenir ?

Dans le cas où un processus a besoin d'un certain nombre de pages, il va vérifier auprès du PageProvider, combien sont disponibles. S'il y en a assez, le processus va se lancer. Problème, au moment de demander les pages en question au PageProvider, celui-ci pourrait les avoir allouées à un autre processus. Résultant en la mort forcée du processus.

Partie 2

Pourquoi ne pas conserver dans la mémoire plusieurs programmes en même temps ?

Il n'est pas viable de conserver plusieurs programmes en même temps dans la mémoire car le risque qu'ils s'endommagent les uns les autres augmente à mesure que le nombre de programmes conservés augmente.

II.7.

Il se peut qu'il râle pour la pile du dernier thread noyau en cours d'exécution, c'est normal, pourquoi ?

C'est normal car chaque thread noyau libère l'espace qu'il a alloué au thread qu'il a créé une fois que ce dernier est terminé. Cependant, il est impossible pour le dernier thread noyau de faire libérer son espace par quoi que ce soit étant donné qu'il est le dernier encore actif.

Limitations

Pour ce dernier rendu, différentes limitations se sont manifestées sur le code final. Tout d'abord, au niveau du forkexec, il est impossible de créer un programme dont la taille du nom dépasse la valeur de MAX_STRING_SIZE (qui vaut 50).
La valeur est définie dans system.h et peut être modifiée à tout moment.

Nous sommes aussi limités au niveau du nombre de processus pouvant s'exécuter en même temps. En effet, notre nombre de pages physiques est limité à 1024. Cela nous limite à 24 processus simultanés avant d'être bloqué (en partant du principe que les processus ne lancent pas de thread).

UserStacksAreaSize a aussi dû être augmenté pour passer à 4096 afin de pouvoir lancer un total de 12 threads par processus.

Cependant, il s'agit de limitations écrites en dur dans le code. Elles peuvent donc être modifiées si besoin.

Choix d'implémentation

Afin de connaître en permanence combien de processus tournent en même temps, nous avons opté pour un compteur présent dans machine.h et machine.cc entouré de sémaphores. De cette façon, nous pouvons empêcher la création de processus si nous n'avons plus la capacité de les créer.

Lorsqu'un processus se termine, et afin d'éviter que la machine s'arrête alors qu'un autre processus est encore en exécution, on vérifie s'il existe un processus encore en cours. La suppression de l'espace d'adressage demande à ce que chaque thread du processus soit terminé afin d'éviter tout un tas de problèmes avec la mémoire. La solution qui nous a semblé la plus simple a été d'ajouter un booléen beDestroyed qui est passé à vrai si le thread est le dernier à tourner.

Nous avons également fait le choix de renvoyer l'appel système Exit à do_forkexit() afin d'éviter que l'un des processus éteigne la machine sans être sûr qu'il s'agit bien du dernier actif.

Afin de mettre à jour le compteur du nombre de processus actifs à un moment donné, la ligne machine->UpdateRunningProcess(1[1]); permet de modifier ce compteur au besoin. Dans progtest.cc (le point de départ du programme de test), nous avons ajouté la ligne ci-dessus afin de bien initialiser le compteur à 1.

[1] 1 pour incrémenter le compteur et -1 pour le décrémenter.