Rapport d'Implémentation pour le Devoir 3 : Pagination dans NACHOS

2023/2024

Magaye Ndiaye

Table des matières

I. Bilan

Le but de ce projet était de mettre en place une gestion de mémoire paginée simple au sein de NACHOS, un pas important vers un modèle de processus similaire à Unix. Ce défi impliquait l'implémentation d'un adressage virtuel par table de pages et la création d'un environnement permettant l'exécution simultanée de plusieurs programmes.

Réalisations :

- Partie I (Adressage virtuel par une table de pages) : Actions I.1 à I.4 ont été complétées avec succès. Cela comprenait l'écriture d'un programme de test, l'examen de l'utilisation de `executable->ReadAt`, la définition d'une fonction `ReadAtVirtual` et la modification de la table des pages pour stresser l'implémentation.
- Partie II (Exécuter plusieurs programmes en même temps) :** Action II.1 a été réalisée, permettant de définir un appel système `ForkExec` pour lancer de nouveaux processus.

Non réalisé :

- Actions I.5 à III n'ont pas été implémentées. Cela inclut la création d'une classe `PageProvider` pour la gestion de l'allocation des pages physiques, les corrections des constructeurs/destructeurs d'`AddrSpace`, et tous les bonus relatifs à la gestion avancée des processus et threads.

Problèmes rencontrés :

- Difficultés liées à la conceptualisation et à l'implémentation des mécanismes de pagination plus complexes.
- Manque de temps pour développer et tester les fonctionnalités plus avancées.

II. Points délicats

L'action la plus complexe réalisée a été l'Action II.1, qui impliquait la création de l'appel système `ForkExec`. Cette fonctionnalité nécessitait une compréhension approfondie de la gestion des processus et de la mémoire dans NACHOS.

Stratégie de résolution :

- Développement d'une approche en deux parties pour `ForkExec`, similaire à la création de thread, mais pour les processus.
- Tests rigoureux pour s'assurer que le nouvel `AddrSpace` était correctement attribué et que les bonnes tables de pages étaient chargées au bon moment.

III. Limitations

Avantages :

- Établissement d'un mécanisme de base pour la pagination et l'exécution simultanée de plusieurs programmes.
- Fondation solide pour des fonctionnalités de gestion de mémoire plus avancées.

Inconvénients :

- Complexité accrue due à la gestion simultanée de multiples processus et espaces d'adressage.
- Manque d'implémentation pour les fonctionnalités avancées, limitant la capacité du système à gérer efficacement la mémoire dans des scénarios complexes.

Limitations principales :

- Absence d'une gestion avancée des pages physiques et de la mémoire, nécessaire pour optimiser l'utilisation des ressources.
- Potentielles difficultés à gérer la terminaison et la synchronisation de multiples processus et threads.

IV. Tests

Des tests ont été effectués pour vérifier les fonctionnalités implémentées, notamment l'adressage virtuel et l'exécution simultanée de programmes.

Stratégie de tests :

- Tests unitaires pour chaque nouvelle fonctionnalité implémentée, comme `ReadAtVirtual` et `ForkExec`.
- Tests d'intégration pour s'assurer que les processus multiples fonctionnaient correctement dans le même espace mémoire.
- Observation des traductions d'adresse et des allocations de mémoire pour identifier d'éventuels problèmes.

Ces tests ont confirmé le fonctionnement de base du système de pagination et de l'exécution simultanée de programmes, tout en soulignant la nécessité d'étendre les capacités de gestion de la mémoire pour des utilisations plus complexes.