Rapport : Implémentation des Entrées/Sorties dans NACHOS

MAHDI Yanis, LUBET Damien

October 6, 2024

Contents

1 Bilan

Dans le cadre de ce projet, nous avons implémenté l'ensemble des fonctionnalités demandées pour la gestion des entrées/sorties dans le système d'exploitation Nachos excepté la récupération de la valeur de retour return n. L'objectif principal était de mettre en place des appels systèmes permettant de gérer les entrées/sorties de manière synchrone et asynchrone. Plusieurs fonctionnalités ont été développées, notamment les appels systèmes pour les fonctions essentielles telles que `PutChar` et `GetChar`, permettant respectivement l'écriture et la lecture de caractères via la console. Nous avons également implémenté des mécanismes de gestion des chaînes de caractères avec l'appel `PutString`. Tous les appels systèmes requis sont fonctionnels, y compris ceux permettant de gérer les caractères en mode utilisateur, avec une gestion correcte des interruptions. Cependant, la dernière partie bonus concernant l'implémentation d'un appel système `printf` n'a pas été réalisée par manque de temps. Mais toutes les autres fonctionnalités essentielles ont

été mises en place avec succès et passent les tests conçus pour valider leur bon fonctionnement.

2 Points délicats

Parmi les aspects les plus complexes que nous avons rencontrés lors de l'implémentation, l'appel système `PutString` a été l'un des plus exigeants. En effet, la gestion des chaînes de caractères posait un défi lorsqu'elles dépassaient la taille maximale du tampon que nous pouvions lire à chaque itération. Pour résoudre ce problème, nous avons dû implémenter une boucle `while` permettant d'écrire la chaîne morceau par morceau, en continuant jusqu'à ce que nous rencontrions le caractère de fin de chaîne (`\0`). Cela garantissait que toutes les chaînes, même les plus longues, étaient correctement écrites, sans provoquer de dépassement de tampon.

Un autre point délicat concernait la gestion de la fonction `main()` et la manière de prendre en compte la valeur de retour `return n`. Nous avons eu des difficultés à comprendre comment récupérer et traiter cette valeur correctement afin de garantir que le programme utilisateur puisse se terminer sans erreurs. On a cependant pas pu trouver la valeur de retour de la fonction `main` malgré le fait que l'on sache qu'elle se trouve dans le registre `R2`.

La gestion de l'appel système `GetString` a également posé problème concernant les débordements notamment coté utilisateur. Nous avons dû faire attention à ne pas consommer plus de caractères que nécessaire, tout en garantissant que les caractères restants soient toujours disponibles dans le flux d'entrée pour d'éventuels appels supplémentaire de la fonction. Cela impliquait de ne pas vider le flux immédiatement, mais seulement lors de la terminaison du programme, afin d'éviter que des caractères indésirables ne soient renvoyés au shell. Pour cela, on a ajouté une variable globale booléenne `checkflux` qui permet de savoir s'il reste des caractères à lire dans le flux d'entrée ou non. Quand c'est le cas, on appelle une fonction `clearStdin` qui vide le flux d'entrée avant de terminer le programme. Cela avoir une fonction getString similaire à la fonction fgets dans son comportement.

3 Limitations

Un des désavantages de notre implémentation est celle de fonction `GetString` qui n'est pas adapté pour l'utilisation de threads. L'appel de la fonction par plusiers threads en même temps peut conduire à des comportement imprévisibles tels que des lectures incorrectes ou des pertes de données. Pour remédier à cela, on aurait pu ajouter des mutex pour garantir l'accès exclusif à la fonction `GetString` à un seul thread à la fois.

4 Tests

Nous avons conçu plusieurs tests pour valider le bon fonctionnement de nos appels systèmes. Le code source des tests contient tous les commentaires nécessaires pour les exécuter avec les bons paramètres et pour connaître la sortie normale attendue à l'écran.

4.1 Stratégie de test pour PutChar

Pour valider le bon fonctionnement de l'appel système `PutChar`, nous avons conçu une série de tests couvrant différents scénarios d'utilisation. L'objectif est de s'assurer que `PutChar` fonctionne correctement dans des conditions variées. Voici les tests effectués :

- **Test de l'écriture d'un seul caractère**
- **Test de l'écriture de plusieurs caractères**
- **Test de l'écriture d'un caractère spécial**

4.2 Stratégie de test pour PutString

Pour valider le bon fonctionnement de l'appel système `PutString`, nous avons conçu une série de tests couvrant différents scénarios d'utilisation. L'objectif est de s'assurer que `PutString` fonctionne correctement dans des conditions variées. Voici les tests effectués :

- **Test de l'écriture d'une chaîne de caractères courte**
- **Test de l'écriture d'une chaîne de caractères longue dépassant la taille du buffer**
- **Test de l'écriture d'une chaîne de caractères contenant un caractère de fin de chaîne au milieu**
- **Test de l'écriture d'une chaîne de caractères vide**

4.3 Stratégie de test pour GetChar

Le test de `GetChar` est assez simple, il suffit de vérifier que la fonction retourne bien le caractère saisi par l'utilisateur. Le programme fonctionne de la manière suivante : il demande à l'utilisateur de saisir un caractère, puis affiche le caractère saisi à l'écran tant que le caractère saisi n'est pas q ou CRTL+D (EOF). Nous avons testé plusieurs cas :

- **Test de la saisie d'un seul caractère**
- **Test de la saisie de plusieurs caractères**
- **Test de la saisie d'un caractère spécial**

4.4 Stratégie de test pour GetString

Le test de `GetString` est similaire à celui de `GetChar`, sauf qu'ici on ne fait pas de boucle. On teste la saisie d'une seule chaîne de caractères par l'utilisateur. Le programme fonctionne de la manière suivante : il demande à l'utilisateur de saisir une chaîne de caractères, puis affiche la chaîne saisie à l'écran. Nous avons testé plusieurs cas :

- **Test de la saisie d'une chaîne de caractères courte**
- **Test de la saisie d'une chaîne de caractères longue** : On teste la saisie d'une chaîne de caractères plus grande que la taille du buffer pour vérifier que la fonction `GetString` récupère correctement le bon nombre de caractères demandé par `getString` sans déborder par la suite dans le shell pour les caractères restants de la chaîne.
- **Test de la saisie d'une chaîne de caractères vide**

4.5 Stratégie de test pour PutInt et GetInt

La stratégie de test pour `PutInt` est similaire à celle de `PutChar` tandis que celle de `GetInt` est similaire à celle de `GetString`.