ENSTA Bretagne
2 rue Francois Verny
29200 Brest Cedex
France
tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr

Eliott Chopin

FISE 2025

20 janvier 2025

Virtualization

Table des matières

1 Introduction **2**

2 Adaptation du Code pour un Module du Noyau **2**

3 Implémentation du DMA **3**

1 Introduction

Maintenant que le Random Number Generator, son driver et l'applications du user-space sont implémentés, ils nous reste à améliorer et à optimiser le driver afin d'obtenir une meilleure modularité ainsi que de meilleures performances.

2 Adaptation du Code pour un Module du Noyau

Dans le cadre de cette tâche, un pilote de générateur de nombres aléatoires (RNG) a été adapté pour fonctionner en tant que module du noyau Linux. L'objectif principal était de permettre la communication entre l'espace utilisateur et l'espace noyau en utilisant des mécanismes du noyau appropriés, tout en assurant une interaction avec le matériel via l'E/S mémoire mappée (MMIO).

Le code a été modifié pour inclure les appels spécifiques au noyau, comme `ioremap`, qui permet de mapper une adresse physique (`DEVICE_BASE_PHYS_ADDR`) dans l'espace virtuel du noyau. Cela permet d'accéder aux registres matériels de manière sécurisée en utilisant `ioread32` pour lire les données et `iowrite32` pour écrire des valeurs, comme la seed du RNG. Pour l'interaction avec l'espace utilisateur, des commandes IOCTL ont été définies : `MY_RNG_IOCTL_RAND` pour générer un nombre aléatoire et `MY_RNG_IOCTL_SEED` pour seeder le générateur.

Lors de l'initialisation du module avec `module_init`, le périphérique a été enregistré à l'aide de `register_chrdev` pour créer une interface de type caractère. Cette interface permet à l'espace utilisateur d'envoyer des commandes IOCTL pour interagir avec le matériel. Lors de la désinstallation du module avec `module_exit`, les ressources sont libérées, et l'accès à la mémoire mappée est annulé avec `iounmap`.

En résumé, cette adaptation permet d'intégrer un pilote de RNG en tant que module noyau, facilitant la gestion des ressources matérielles et offrant une interface efficace pour l'espace utilisateur grâce aux mécanismes du noyau Linux.

Résolution de la valeure de base hardcodée

Le module du noyau utilisé pour interagir avec un périphérique PCI de type générateur de nombres aléatoires (RNG) avait une adresse de base codée en dur. Cela posait des problèmes de flexibilité, car l'adresse de base pouvait varier lors du redémarrage du système ou de modifications matérielles. Il était également nécessaire pour l'utilisateur de renseigner manuellement cette adresse lors du chargement du module.

Solution

Pour résoudre ce problème, nous avons modifié le module du noyau pour qu'il détecte automatiquement l'adresse de base du périphérique PCI au moment du chargement. La solution repose sur l'utilisation de la fonction `pci_get_device()`, qui permet de rechercher un périphérique PCI par son identifiant de vendeur et de périphérique. Une fois le périphérique trouvé, nous accédons à l'adresse de base à travers la ressource mémoire du périphérique et la mappage avec `ioremap()`.

Étapes de l'implémentation

L'implémentation a débuté par l'identification du périphérique PCI, une étape réalisée grâce à la fonction `pci_get_device()`, permettant de localiser le matériel en fonction de ses identifiants uniques de vendeur et

de périphérique. Une fois le périphérique détecté, son adresse de base a été extraite à partir de ses ressources mémoire, spécifiquement via le champ `pdev->resource[0]`.

Pour accéder efficacement à la mémoire du périphérique, cette adresse physique a été mappée en mémoire virtuelle à l'aide de `ioremap()`, facilitant ainsi les opérations en espace noyau. Enfin, le module du noyau a été enregistré pour activer la communication avec le périphérique, en veillant à libérer les ressources correctement lors de son déchargement. Cette approche garantit robustesse et stabilité dans l'interaction avec le matériel.

Avantages de la solution

La détection de l'adresse de base se fait automatiquement au moment du chargement du module, éliminant le besoin d'une configuration manuelle. Cette solution permet d'éviter les erreurs humaines et améliore la portabilité du module. Elle rend également le système plus flexible en permettant une adaptation dynamique aux changements matériels sans nécessiter de modifications du code ou de paramètres de l'utilisateur.

3 Implémentation du DMA

Le projet visait à améliorer les performances de génération de nombres aléatoires d'un périphérique virtuel, en réduisant la latence liée aux traversées entre l'espace utilisateur et l'espace noyau. Cette amélioration a été obtenue en utilisant le transfert Direct Memory Access (DMA). En effet, les appels fréquents au noyau pour la gestion des entrées-sorties ralentissaient le débit de génération. Le DMA permet de transférer des données de manière plus efficace, directement entre le périphérique et la mémoire, sans nécessiter l'intervention du processeur. Cela réduit la latence et améliore l'efficacité globale du système.

L'implémentation a été réalisée en configurant les registres du périphérique EDU, notamment les registres DMA source, destination et de comptage. Un transfert de données a été initié à l'aide du registre de commande DMA. Ce processus permet de déplacer des blocs de données de manière plus rapide et efficace, en libérant le processeur des tâches de copie et en augmentant ainsi le débit de génération des nombres aléatoires. Cette optimisation a permis une amélioration significative des performances du périphérique dans le contexte de la génération de données aléatoires à haut débit.