CRÉATION DE SÉPARATION LOGICIEL POUR MIXTE SYSTÈMES DE CRITICITÉ

ANDREW CAPLES, RESPONSABLE MARKETING PRODUITS NUCLEUS
WAQAR SADIQ, INGÉNIEUR TECHNIQUE MARKETING

Mentor®
A Siemens Business

BIVARNEC

INTRODUCTION

L'introduction de puissants processeurs intégrés qui ont conduit à la consolidation des systèmes pour les appareils embarqués non liés à la sécurité conduit désormais à la consolidation des systèmes pour les appareils critiques en matière de sécurité. Alors que les logiciels deviennent de plus en plus complexes à mesure que davantage de fonctionnalités sont fusionnées sur un seul système sur puce (SoC), les fabricants d'appareils sont confrontés à une surveillance accrue de la part des agences de réglementation quant à la sécurité de leurs appareils. Dans un système sûr, les logiciels critiques pour la sécurité doivent disposer d'un accès garanti et prévisible aux ressources de calcul et aux autres ressources du système. Dans un système à criticité mixte, un accès garanti et prévisible doit exister pour les composants critiques pour la sécurité.

Ce mélange de logiciels critiques et non critiques pour la sécurité est possible sur les processeurs modernes d'aujourd'hui, mais ajoute à la complexité globale de la conception. Pour garantir l'accès aux ressources par l'application de sécurité, il doit y avoir une isolation du code non lié à la sécurité pour éviter toute interférence ; et ainsi, plusieurs domaines clés doivent être pris en compte, notamment le partitionnement de la mémoire et la gestion de l'accès aux ressources système.

RÉGIONS DE MÉMOIRE

De manière générale, la mémoire système est partitionnée dans les périphériques intégrés en stockage statique et dynamique (Figure 1). Le stockage statique est l'endroit où résident le « texte » et les « données » ; tandis que le stockage dynamique contient de la mémoire pour la pile (le RTOS et chaque thread RTOS auront sa propre pile) et le tas. Pour les appareils architecturés avec une carte mémoire linéaire, des problèmes de fiabilité surviennent lorsque les sous-systèmes logiciels dépassent leur plage de mémoire désignée dans la mémoire désignée pour d'autres modules. Cette portée excessive peut transcender la mémoire allouée à l'application pour inclure l'espace mémoire alloué au noyau RTOS. Étant donné que l'intégralité de l'espace mémoire est vue par tous les modules logiciels, un sous-système suite à une écriture mémoire erronée peut potentiellement mettre hors service l'ensemble du périphérique.

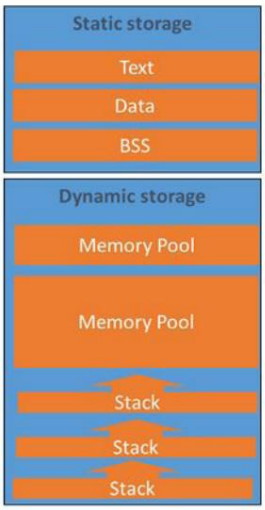

Figure 1 : La différence entre le stockage statique et dynamique. Quel que soit le type de stockage, il est impératif d'isoler correctement la mémoire dans chaque sous-système respectif.

Par conséquent, une considération importante dans les dispositifs de sécurité est l'isolation de la mémoire qui garantit que chaque sous-système est confiné à son propre conteneur de mémoire désigné.

PARTITIONNEMENT DE LA MÉMOIRE

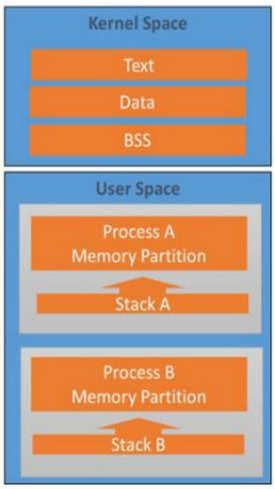

Figure 2 : Partitionnement de la mémoire avec des processus pour créer des régions protégées par la mémoire.

La capacité de partitionner la mémoire en régions protégées, isolées des autres sous-systèmes logiciels, établit une base pour construire des systèmes à criticité mixte composés de modules logiciels de sécurité et non liés à la sécurité. Le partitionnement de la mémoire (Figure 2) crée des domaines d'espace avec des privilèges d'accès et de lecture/écriture définis qui empêchent un sous-système (ou un processus) d'accéder à une mémoire autre que celle qui lui a été spécifiquement allouée. Par exemple, les pointeurs errants dans un processus non sécurisé sont contenus dans leur domaine mémoire respectif et ne peuvent pas avoir d'impact sur la mémoire affectée à un processus de sécurité. En conséquence, le partitionnement de l'espace augmente la fiabilité du système en contenant les défauts du processus, ce qui protège les autres processus et le système dans son ensemble.
La récupération après une panne peut ainsi être réduite du simple redémarrage de l'appareil à quelque chose de plus gérable, comme le simple redémarrage ou le rechargement d'un module de processus.

Une solution est un RTOS avec un modèle de processus léger (tel que le RTOS Nucleus de Mentor) ce qui évite les frais généraux pour maintenir les performances du système en temps réel et peut être utilisé sur tout le spectre des processeurs intégrés modernes, des MCU (microcontrôleurs) aux MPU (microprocesseurs).

ACCÈS AU SYSTÈME

L'isolation fournie par un modèle de processus léger permet une séparation spatiale entre les processus critiques et non critiques pour la sécurité, mais la séparation spatiale en elle-même ne va pas assez loin car il n'y a pas de séparation temporelle. Dans ce cas, les processus de sécurité et ceux qui ne le sont pas ont un accès complet aux ressources du noyau. À titre d'exemple, considérons un processus non lié à la sécurité qui tente de transmettre en continu des données sur un réseau local (LAN) et l'effet sur les ressources système qui peut être requis par le processus de sécurité (Figure 3).

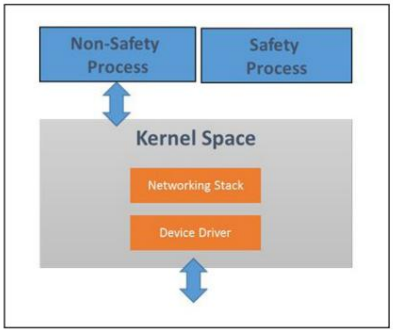

Figure 3 : Un processus non sécurisé qui accède en permanence à l'espace du noyau, mais partage les services du noyau avec le processus de sécurité.

Comme le montre la figure 3, même si le processus de sécurité de priorité plus élevée peut préempter le processus non sécurisé, un pilote de périphérique réseau pourrait être écrit de telle sorte qu'il y ait des périodes pendant la transmission des paquets au cours desquelles les interruptions sont désactivées pour les sections critiques du code, ou non réentrantes. code, qui empêche intrinsèquement le processus de sécurité d'accéder aux ressources du système. Le résultat net est qu'il serait possible que le processus non lié à la sécurité bloque un processus lié à la sécurité, affectant ainsi le caractère sacré d'une réponse déterministe. Sans mécanisme intégré pour limiter l'accès au système, les ressources système pourraient continuer à être consommées par un processus non lié à la sécurité. Pour les appareils critiques pour la sécurité, toute entrave aux ressources système par un code non critique pour la sécurité est inacceptable. Afin de garantir que le code lié à la sécurité s'exécute de manière prévisible, il est nécessaire de gérer l'accès aux ressources du système.

Heureusement, avec un modèle de processus léger, il existe des solutions pour empêcher un accès illimité aux ressources système : par exemple, déplacer les pilotes de périphérique vers l'espace utilisateur et déplacer le code critique pour la sécurité vers l'espace du noyau.

MIDDLEWARE DE L'ESPACE UTILISATEUR ET PILOTES DE PÉRIPHÉRIQUES

Middleware/ Device Drivers	Middleware/Device Separation	Interrupts	Impact
Kernel Space	Drivers are not isolated. Accessible by safe and non-safe processes	Drivers can disable interrupts	Non-safe process can make unfettered system calls and impact system performance and resource availability
User Space	Drivers are isolated. Processes in User Space provide isolation	Drivers cannot disable interrupts	Throttling of system calls by non-safe processes can be managed by polling

Figure 4 : Comprendre les comportements du noyau et de l'espace utilisateur.

Pour les systèmes à criticité mixte, il est nécessaire de garantir qu'un code non lié à la sécurité ne puisse pas avoir d'impact sur la réponse déterministe du processus de sécurité. Étant donné que les pilotes de périphériques et les middlewares résident généralement dans l'espace du noyau, ils ont accès aux API du noyau, y compris celles qui peuvent être utilisées pour désactiver les interruptions susceptibles de bloquer le code critique pour la sécurité.

Pour contourner ce problème, Nucleus Process Model permet de déplacer les pilotes de périphériques et les middlewares vers des régions protégées de l'espace utilisateur. Le code exécuté dans l'espace utilisateur n'aura accès qu'à un sous-ensemble d'API du noyau. Et par conséquent, le code ne pourra pas modifier le domaine temporel. Ainsi, les processus de sécurité, lorsqu'ils seront prêts à être exécutés, pourront accéder aux ressources du système de manière prévisible et déterministe. Si un processus non sécurisé tente de consommer des ressources système pour des tâches d'E/S gourmandes en calcul (via l'utilisation d'appels d'interrogation), le noyau peut gérer la quantité d'utilisation des ressources fournie (Figure 4). Pendant les périodes pendant lesquelles le code de sécurité est inactif, le noyau peut utiliser les cycles disponibles pour interroger les requêtes d'E/S. Étant donné que le processus de sécurité aura la priorité la plus élevée, toute demande de ressource système aura priorité sur le code non lié à la sécurité pour garantir une réponse déterministe.

CODE CRITIQUE POUR LA SÉCURITÉ DANS L'ESPACE DU NOYAU

Pour fournir une plus grande séparation spatiale et temporelle, le code critique pour la sécurité peut être déplacé vers l'espace du noyau. Cette architecture logicielle peut être qualifiée de mode « premier plan/arrière-plan ». Le processus de sécurité est considéré comme le mode de premier plan s'exécutant dans l'espace du noyau avec la priorité la plus élevée ; tandis que l'arrière-plan est constitué de processus en mode utilisateur composés de codes non liés à la sécurité. Les processus non sécurisés fonctionnant en mode arrière-plan ne sont pas en mesure d'affecter les processus de sécurité en mode premier plan – ni spatialement (en raison de la protection de la mémoire fournie par le modèle de processus) ni temporellement (en limitant l'accès aux API du noyau utilisées pour la désactivation des interruptions).).

À titre d'exemple, des tests de performances visant à déterminer la réponse critique pour la sécurité ont été mesurés avec un système d'exploitation commercial comprenant un modèle de processus léger et examinés dans différentes conditions de charge. L'environnement utilisé était Mentor's Nucleus RTOS (v2017.2) sur un SabreLite NXP i.MX6 fonctionnant à 998 MHz, dans lequel le processeur, le noyau et le pilote ont été chargés comme suit :

1. Charge CPU : une tâche de priorité moyenne incrémentant un compteur dans une boucle while (1).
2. Charge du noyau : deux tâches de priorité moyenne synchronisées via un sémaphore utilisant while (1) boucle dans laquelle une tâche libère le sémaphore tandis que l'autre tâche parvient à charger complètement le noyau et le planificateur.
3. Charge du pilote : une tâche de priorité moyenne envoie des paquets UDP de 1 024 octets en continu, ce qui génère plusieurs interruptions dans le système pour l'accès Ethernet.

	Modules Loading the System Executing as User Processes					
CPU Load	Kernel Load	Driver Load	Interrupt Latency		Schedule Latency	
			Average	Range	Average	Range
Disabled	Disabled	Disabled	0.7 µsec	0.6-0.8	1.4 µsec	1.2-1.8
Enabled	Disabled	Disabled	0.7 µsec	0.6-0.8	1.4 µsec	1.2-1.8
Enabled	Enabled	Disabled	0.7 µsec	0.6-0.8	1.5 µsec	1.4-1.8
Enabled	Enabled	Enabled	0.7 µsec	0.6-3.0	2.2 µsec	1.4-15.0

Figure 5 : Latence d'interruption : temps écoulé entre l'apparition d'une interruption de ligne externe et la première instruction exécutée dans l'ISR enregistré.
Latence de planification : temps écoulé entre le gestionnaire d'interruption et l'exécution de la première instruction dans la tâche en attente la plus prioritaire.

En regardant la figure 5, remarquez que les temps de réponse pour la latence d'interruption et de planification restent cohérents pendant les tests dans lesquels le processeur et le noyau sont chargés. Cependant, une fois le pilote réseau chargé, les variations d'interruption ont augmenté. Notez également la latence de planification, qui augmente dans des conditions de charge du pilote. En apparence, il semble que le pilote réseau ayant accès aux API privilégiées désactive les interruptions pour dégrader les temps de réponse car il consomme les cycles du processeur.

Le déplacement du pilote de périphérique vers un processus utilisateur doté d'un mécanisme d'interrogation produit les résultats suivants pour le même environnement de test.

	Modules Loading the System Executing as User Processes and the Ethernet Driver Process in Poll Mode					
CPU Load	Kernel Load	Driver Load	Interrupt Latency		Schedule Latency	
			Average	Range	Average	Range
Disabled	Disabled	Disabled	0.7 µsec	0.6-0.8	1.4 µsec	1.2-1.8 usec
Enabled	Disabled	Disabled	0.7 µsec	0.6-0.8	1.4 µsec	1.2-1.8 usec
Enabled	Enabled	Disabled	0.7 µsec	0.6-0.8	1.5 µsec	1.4-1.8 usec
Enabled	Enabled	Enabled	0.7 µsec	0.6-0.8	1.5 µsec	1.4-1.8 usec

Figure 6 : Latence d'interruption : temps écoulé entre l'apparition d'une interruption de ligne externe et la première instruction exécutée dans l'ISR enregistré.
Latence de planification : temps écoulé entre le gestionnaire d'interruption et l'exécution de la première instruction dans la tâche en attente la plus prioritaire.

Dans la figure 6, avec le pilote dans un processus utilisateur, ni la latence d'interruption ni la latence de planification ne sont affectées lorsque l'application non sécurisée tente de transmettre en continu des paquets UDP. En effet, le pilote réseau peut accéder aux services du noyau, mais n'a pas accès aux API privilégiées pour désactiver les interruptions. Le service est fourni à intervalles contrôlés. Le résultat net est que le module certifié en matière de sécurité n'est pas affecté et peut continuer à répondre aux exigences du système.

UN MODÈLE DE PROCÉDÉ ÉPROUVÉ POUR LES SYSTÈMES À CRITICITÉ MIXTE

Nucleus RTOS inclut un modèle de processus léger (Figure 7) qui exploite l'unité de gestion de la mémoire (MMU) ou l'unité de protection de la mémoire (MPU) sur le processeur pour appliquer les politiques de lecture/écriture pour chaque région de mémoire.
Puisqu'il s'agit d'un modèle de processus léger, il n'est pas nécessaire de virtualiser la mémoire. L'avantage de cette démarche est de réduire les frais généraux inutiles qui peuvent avoir un impact sur les performances. Les développeurs de logiciels et les architectes système peuvent donc déployer un système avec une carte mémoire plate et linéaire avec des régions de mémoire protégées pour l'espace utilisateur et le noyau. Ce type d'approche isole également les processus individuels dans l'espace utilisateur et isole à la fois le noyau et l'espace utilisateur. Pour les modules critiques pour la sécurité, les processus peuvent être utilisés pour isoler les processus non critiques pour la sécurité afin de créer des dispositifs certifiables qui répondent aux plus hauts niveaux des normes de sécurité ISO et CEI.

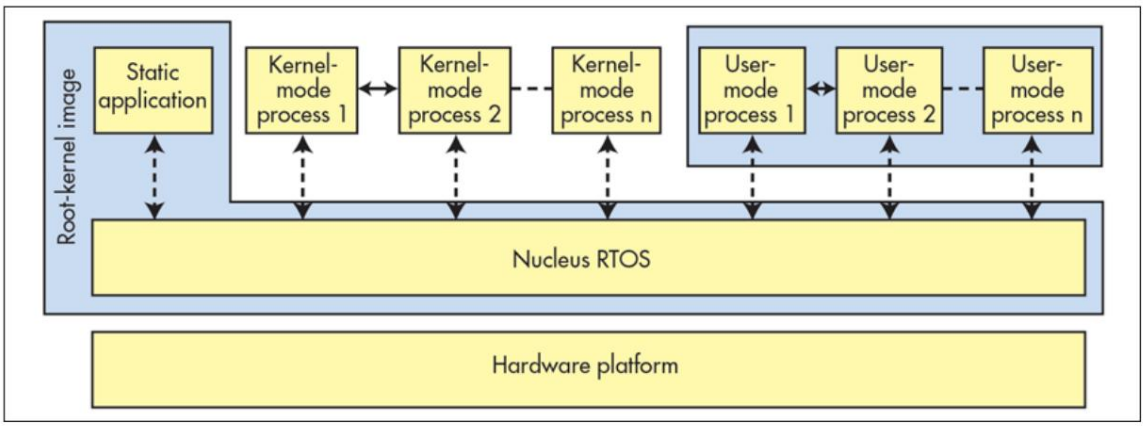

Figure 7 : Le modèle de processus Nucleus est une approche légère du partitionnement de l'espace qui crée des régions de mémoire protégées.

CONCLUSION

L'utilisation d'un modèle de processus léger pour la séparation peut fournir l'isolation requise pour les conceptions à criticité mixte. Grâce à l'utilisation du partitionnement de la mémoire, une séparation spatiale est obtenue pour créer des domaines spatiaux avec des niveaux de privilèges d'accès attribués qui servent à contenir les pannes des processus individuels, ce qui augmente la fiabilité du système.

La séparation temporelle peut être obtenue grâce à l'utilisation d'une implémentation en arrière-plan/avant-plan qui garantit l'accès aux ressources système au code critique pour la sécurité s'exécutant dans l'espace du noyau en tant que processus de premier plan.
Pour de nombreuses conceptions, l'utilisation d'un modèle de processus léger peut fournir l'isolation requise pour répondre au plus haut niveau de certification de sécurité tout en conservant les réponses en temps réel nécessaires pour répondre aux exigences de performances les plus exigeantes.

Visitez la page des processeurs pris en charge par Mentor's Nucleus pour voir si Nucleus prend en charge votre processeur actuel.

Biographies des auteurs :

Andrew Caples est responsable du marketing produit pour la division Systèmes embarqués (ESD) chez Mentor. Il possède plus de 20 ans d'expérience dans des start-ups et des entreprises de haute technologie classées Fortune 500 et a occupé divers postes allant du marketing technique à la gestion des ventes. Il est titulaire d'un baccalauréat en génie électrique et informatique de l'Université polytechnique de Californie. Ses responsabilités actuelles incluent la gestion de produits pour le système d'exploitation en temps réel Nucleus.

Waqar Sadiq est ingénieur technique marketing senior pour la division Systèmes embarqués (ESD) chez Mentor. Il travaille actuellement sur le système d'exploitation Nucleus Real-Time et sur d'autres technologies d'exécution. Waqar a passé plus de 17 ans dans l'industrie électronique, travaillant à la fois dans la conception et le développement de matériel et de logiciels. Il a occupé divers postes en développement et en gestion avant d'accéder à son rôle actuel chez Mentor. Waqar est titulaire d'un BS en électronique et d'une maîtrise en génie informatique.

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Trust Tower
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

MGC 11-17 TECH16320-w