Formation Linux, drivers et programmation noyau

N° ACTIVITÉ : 11 92 18558 92 | TÉLÉPHONE : 01 85 77 07 07 | E-MAIL : inscription@hubformation.com

Cette formation vous permettra de maîtriser le développement de pilotes de périphériques (drivers) robustes et adaptés aux différentes distributions de Linux. Vous verrez les différents types de périphériques, la gestion de la mémoire, l'implémentation de protocole réseau ainsi que les périphériques USB.

Référence	LDI
Durée	4 jours (28h)
Tarif	2 450 €HT
Repas	repas inclus

Objectifs

| Maîtriser le développement de pilotes de périphériques
| Identifier en détail les mécanismes internes du noyau
| Savoir développer et intégrer de nouveaux éléments dans le noyau Linux
| Ecrire un pilote périphérique en mode caractère ou bloc

Public

| Développeurs Linux/Unix

Prérequis

| Bonnes connaissances de Linux/Unix et de la programmation C

Programme de la formation

Présentation du noyau
| Vue d'ensemble du système et rôle du noyau.
| Les sites de référence.
| Spécificités des noyaux 3.x et 4.x.
| Cycles de développement du noyau, les patchs.
| Mode de fonctionnement (superviseur et utilisateur). Appels système.
| Organisation des sources (Include/linux, Arch, Kernel, Documentation...).
| Principe de compilation du noyau et des modules.
| Les dépendances et symboles.
| Les exportations de symboles.
| Le chargement du noyau (support, argument...).
| Travaux pratiques Compilation et installation d'un noyau 3.x.

Les outils utilisables
| Outils de développement (Gcc, Kbuild, Kconfig et Makefile...).
| Outils de débogage (GDB, KGDB, ftrace...).
| Environnement de débogage (Linux Trace Toolkit...).
| Outil de gestion de version (Git...).
| Tracer les appels système (ptrace...).
| Travaux pratiques Installer l'ensemble des outils et des sources pour générer un module pour le noyau. Configurer le système pour effectuer le chargement automatique de module au boot. Ecriture et test de modules simples.

Gestion des threads, scheduling
| Les différents types de périphériques.
| Contextes de fonctionnement du noyau. Protection des variables globales.
| Représentation des threads (état, structure task_stru, thread_info...).
| Les threads, contexte d'exécution.
| Le scheduler de Linux et la préemption.
| Création d'un thread noyau (kthread_create, wakeup_process...).

SESSIONS PROGRAMMÉES

A DISTANCE (FRA)
du 1er au 4 juillet 2025
du 23 au 26 septembre 2025
du 9 au 12 décembre 2025

PARIS
du 16 au 19 septembre 2025
du 2 au 5 décembre 2025

VOIR TOUTES LES DATES

| Travaux pratiques Créer un module qui crée un thread noyau lors de l'insertion et le décharge lors du rmmod. Ecriture d'un module d'horodatage d'événements à haute précision. Ecriture d'un module d'information sur les structures internes des processus.

Gestion de la mémoire, du temps et de proc
| L'organisation mémoire pour les architectures UMA et NUMA.
| L'espace d'adressage utilisateur et noyau.
| La gestion de pages à la demande (demand paging).
| Allocations mémoire, buddy allocator, kmalloc, slabs et pools mémoire.
| La gestion des accès à la mémoire (les caches et la MMU).
| Les problèmes liés à la sur-réservation de la mémoire.
| Gestion de la mémoire sur x86 et ARM, utilisation des Hugepages.
| Optimisation des appels systèmes (IAPX32, VDSO).
| Synchronisations et attentes dans le noyau, waitqueues, mutex et les completions.
| Les ticks et les jiffies dans Linux.
| L'horloge temps réel, RTC (real Time Clock), implémentation des timers.
| Interface timers haute résolution, estampilles.
| Les outils spécifiques au noyau, listes chaînées, kfifo et container_of.
| L'interface noyau avec /proc par le procfs.
| Travaux pratiques Utilisation des timers et des estampilles. Implémentation d'un accès au procfs. Mise en oeuvre de l'allocation mémoire dans le noyau et optimisation à l'aide des slabs.

Périphérique en mode caractère
| Ecriture de pilotes de périphériques caractère.
| Le VFS (Virtual File System).
| Les méthodes associées aux périphériques caractères.
| Gestion des interruptions DMA et accès au matériel.
| Enregistrement des pilotes de périphériques de type caractère et optimisations.
| Travaux pratiques Ecriture progressive d'un pilote périphérique en mode caractère. Implémentation des synchronisations d'entrée-sortie entre threads et avec la routine d'interruption. Implémentation de l'allocation mémoire.

Linux Driver Framework - sysfs
| Présentation du framework, kobject, kset et kref.
| Les objets drivers, device driver, bus et class.
| Utilisation et génération des attributs présentés dans le sysfs.
| Interface avec le hotplug, méthodes match, probe et release.
| Gestion du firmware.
| Gestion de l'énergie, méthodes de gestion de l'énergie.
| Travaux pratiques Implémentation d'un bus, d'un driver et d'un device driver. Adaptation du pilote de périphériques caractère. Exemple d'utilisation de l'interface.

Périphérique en mode bloc et systèmes de fichiers
| Principe des périphériques en mode bloc. Enregistrement du driver.
| Callback de lecture et écriture. Support du formatage et opérations avancées.
| Ordonnanceur des entrées-sorties par bloc du noyau.
| Conception des systèmes de fichiers.
| Enregistrement d'un nouveau système de fichiers.
| Travaux pratiques Exemple de pilote complet de périphérique virtuel. Exemple d'un système de fichiers personnalisé.

Interfaces et protocoles réseau
| Gestion des interfaces réseau sous Linux.
| Utilisation des skbuff.
| Les hooks netfilter.
| Intégration d'un protocole.
| Travaux pratiques Exemple de driver réseau pour périphérique virtuel. Implémentation de protocole réseau.

Drivers pour périphériques USB
| Principe des périphériques USB. Interface avec le module USB-core.
| Interaction du périphérique avec le noyau Linux.
| Construction d'un URB (USB Request Block).
| Les gadgets USB.
| Travaux pratiques Enregistrement d'un driver USB. Ecriture d'un driver en mode isochrone.

Méthode pédagogique

Les nombreux exercices et études de cas progressifs seront réalisés sur un réseau de serveurs Linux. Tous les programmes réalisés en TP existent sous forme de squelettes que les participants complètent eux-mêmes.

Méthode d'évaluation

Tout au long de la formation, les exercices et mises en situation permettent de valider et contrôler les acquis du stagiaire. En fin de formation, le stagiaire complète un QCM d'auto-évaluation.

Suivre cette formation à distance

Voici les prérequis techniques pour pouvoir suivre le cours à distance :

| Un ordinateur avec webcam, micro, haut-parleur et un navigateur (de préférence Chrome ou Firefox). Un casque n'est pas nécessaire suivant l'environnement.
| Une connexion Internet de type ADSL ou supérieure. Attention, une connexion Internet ne permettant pas, par exemple, de recevoir la télévision par Internet, ne sera pas suffisante, cela engendrera des déconnexions intempestives du stagiaire et dérangera toute la classe.
| Privilégier une connexion filaire plutôt que le Wifi.
| Avoir accès au poste depuis lequel vous suivrez le cours à distance au moins 2 jours avant la formation pour effectuer les tests de connexion préalables.
| Votre numéro de téléphone portable (pour l'envoi du mot de passe d'accès aux supports de cours et pour une messagerie instantanée autre que celle intégrée à la classe virtuelle).
| Selon la formation, une configuration spécifique de votre machine peut être attendue, merci de nous contacter.
| Pour les formations incluant le passage d'une certification la dernière journée, un voucher vous est fourni pour passer l'examen en ligne.
| Pour les formations logiciel (Adobe, Microsoft Office...), il est nécessaire d'avoir le logiciel installé sur votre machine, nous ne fournissons pas de licence ou de version test.
| Horaires identiques au présentiel.

Accessibilité

Les sessions de formation se déroulent sur des sites différents selon les villes ou les dates, merci de nous contacter pour vérifier l'accessibilité aux personnes à mobilité réduite.
Pour tout besoin spécifique (vue, audition...), veuillez nous contacter au 01 85 77 07 07.