+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Mis à jour le 23/01/2026

S'inscrire

Formation développement Linux embarqué

4 jours (28 heures)

Présentation

Le développement de pilotes (drivers) sous Linux est une compétence critique pour l'industrie, où la stabilité du système repose sur la qualité du code bas niveau. Contrairement au développement applicatif, une erreur dans le noyau ne pardonne pas. Il est donc essentiel de maîtriser les mécanismes internes du Kernel pour intégrer correctement de nouveaux périphériques.

Notre formation démystifie le fonctionnement du noyau Linux. Vous apprendrez à naviguer dans les sources, à configurer et compiler un noyau sur mesure, et surtout à écrire vos propres modules et pilotes de périphériques. Nous insisterons particulièrement sur les standards modernes comme le Device Tree, la gestion sécurisée de la mémoire et les mécanismes de synchronisation (Locking), indispensables pour garantir la robustesse de vos systèmes embarqués.

À l'issue de la formation, vous serez autonome pour écrire un driver complet, gérer les interruptions matérielles et déboguer efficacement vos modules noyau.

Objectifs

- Comprendre l'architecture et les sous-systèmes du noyau Linux.
- Configurer (Kconfig) et compiler (Kbuild) un noyau personnalisé.
- Développer et intégrer des modules noyau (LKM).
- Écrire un pilote de périphérique caractère (Character Driver).
- Maîtriser la syntaxe du Device Tree pour décrire le matériel.
- Gérer les interruptions et les accès mémoire (I/O).

Public visé

- Développeurs C Bas niveau / Embarqué
- Ingénieurs Système Linux

Pré-requis

- Maîtrise solide du langage C (pointeurs, structures, gestion mémoire).
- Aisance avec les commandes Linux et le Shell.
- La connaissance de l'architecture matériel (processeur, bus) est un plus.

Pré-requis techniques

- PC Linux ou Machine Virtuelle fournie.
- Cible matérielle : QEMU (émulation) et/ou Raspberry Pi pour les tests réels.

Formation développement Linux embarqué

[Jour 1 - Matin]

Architecture et Build System

- Espace Utilisateur vs Espace Noyau (User space vs Kernel space)
- Exploration des sources du noyau Linux (Mainline)
- Le système de configuration Kconfig
- Le système de build Kbuild et Makefiles
- Atelier pratique : Configuration, compilation et boot d'un noyau sur mesure sur QEMU.

[Jour 1 - Après-midi]

Premier Module Noyau

- Structure d'un module (Loadable Kernel Module)
- Macros
- Compilation "Out-of-tree"
- Gestion des licences et métadonnées
- Commandes
- Atelier pratique : Développement, compilation et test du module "Hello Kernel".

[Jour 2 - Matin]

Pilotes de Caractères (Char Drivers)

- Concept de fichiers de périphériques (/dev)
- Numéros majeurs et mineurs
- La structure

- Implémentation des fonctions
- Atelier pratique : Création d'un driver simple et enregistrement du périphérique.

[Jour 2 - Après-midi]

Échange avec l'User Space

- Ségrégation mémoire Kernel/User
- Fonctions de transfert
- Validation des pointeurs et sécurité
- Utilisation de commandes spécifiques
- Atelier pratique : Pilotage du driver depuis une application utilisateur en C.

[Jour 3 - Matin]

Device Tree et Platform Drivers

- Le rôle du Device Tree (DTS, DTB, DTC)
- Décrire le hardware non-découvrable
- Le modèle "Platform Driver" : Probe et Remove
- Matching via "Compatible strings"
- Atelier pratique : Modification du Device Tree pour déclarer un périphérique virtuel personnalisé.

[Jour 3 - Après-midi]

Accès au Matériel (I/O Memory)

- Architecture mémoire et I/O Mapping
- Allocation de ressources
- Mapping virtuel
- Primitives d'accès registres
- Atelier pratique : Pilotage de LED/GPIO via manipulation directe des registres.

[Jour 4 - Matin]

Gestion des Interruptions

- Fonctionnement des IRQ sous Linux
- Enregistrement d'un handler
- Contraintes contextuelles (pas de sommeil en interruption)
- Traitement différé (Bottom Halves) : Tasklets et Workqueues
- Atelier pratique : Gestion d'une interruption simulée (ex: bouton poussoir).

[Jour 4 - Après-midi]

Concurrence et Debugging Avancé

- Comprendre les Race Conditions et la réentrance
- Primitives de verrouillage : Mutex vs Spinlocks
- Outils de debugging
- Analyse de "Kernel Panic" et "Oops"
- Atelier pratique : Sécuriser le driver avec des Mutex et session de debug d'un crash.

Sociétés concernées

Cette formation s’adresse à la fois aux particuliers ainsi qu’aux entreprises, petites ou grandes, souhaitant former ses équipes à une nouvelle technologie informatique avancée ou bien à acquérir des connaissances métiers spécifiques ou des méthodes modernes.

Positionnement à l'entrée en formation

Le positionnement à l'entrée en formation respecte les critères qualité Qualiopi. Dès son inscription définitive, l'apprenant reçoit un questionnaire d’auto-évaluation nous permettant d’apprécier son niveau estimé sur différents types de technologies, ses attentes et objectifs personnels quant à la formation à venir, dans les limites imposées par le format sélectionné. Ce questionnaire nous permet également d’anticiper certaines difficultés de connexion ou de sécurité interne en entreprise (intraentreprise ou classe virtuelle) qui pourraient être problématiques pour le suivi et le bon déroulement de la session de formation.

Méthodes pédagogiques

Stage Pratique : 60% Pratique, 40% Théorie. Support de la formation distribué au format numérique à tous les participants.

Organisation

Le cours alterne les apports théoriques du formateur soutenus par des exemples et des séances de réflexions, et de travail en groupe.

Validation

À la fin de la session, un questionnaire à choix multiples permet de vérifier l’acquisition correcte des compétences.

Sanction

Une attestation sera remise à chaque stagiaire qui aura suivi la totalité de la formation.