



Formation optimisation du temps de démarrage de Linux embarqué

Formation sur site, 3 jours

Dernière mise à jour : 24 décembre 2025

Titre	Formation optimisation du temps de démarrage de Linux embarqué
Objectifs opérationnels	<ul style="list-style-type: none">• Être capable d'utiliser les outils et techniques pour mesurer le temps de démarrage d'un système embarqué.• Être capable de réduire le temps de démarrage au niveau de l'initialisation de l'espace utilisateur Linux.• Être capable de réduire le temps de démarrage au niveau de l'initialisation du noyau Linux.• Être capable de réduire le temps de démarrage au niveau de l'initialisation du chargeur de démarrage.• Être capable de mettre en oeuvre d'autres techniques avancées et alternatives d'optimisation du temps de démarrage.
Durée	Trois jours - 24 h (8 h par jour)
Méthodes pédagogiques	<ul style="list-style-type: none">• Présentations animées par le formateur : 40% de la durée de formation• Travaux pratiques réalisés par les participants : 60% de la durée de formation• Version électroniques de supports de présentation, des instructions et des données de travaux pratiques. Les supports sont librement disponibles sur https://bootlin.com/doc/training/boot-time.
Formateur	Un des ingénieurs mentionnés sur : https://bootlin.com/training/trainers/
Langue	Présentations : Français Supports : Anglais
Public visé	Sociétés et ingénieurs développeurs de systèmes Linux embarqués. Personnes offrant de l'assistance à de tels développeurs.



Pré-requis

- **Connaissance et pratique des commandes UNIX ou GNU/Linux :** les participants doivent être à l'aise avec l'utilisation de la ligne de commande Linux. Les participants manquant d'expérience sur ce sujet doivent se former par eux-mêmes, par exemple en utilisant nos supports de formation disponible à l'adresse bootlin.com/blog/command-line/.
- **Expérience minimale en développement Linux embarqué :** les participants doivent avoir une compréhension minimale de l'architecture d'un système Linux embarqué : rôle du noyau Linux par rapport à l'espace utilisateur, développement d'applications espace utilisateur en C. Suivre la formation *Linux embarqué* de Bootlin, disponible sur bootlin.com/training/embedded-linux/, permet de remplir ce pré-requis.
- **Niveau minimal requis en anglais : B1**, d'après le *Common European Framework of References for Languages*, pour nos sessions animées en anglais. Voir bootlin.com/pub/training/cefr-grid.pdf pour une auto-évaluation.

Équipement nécessaire

Pour les sessions en présentiel dans les locaux de nos clients, notre client doit fournir :

- Projecteur vidéo
- Un ordinateur sur chaque bureau (pour une ou deux personnes), avec au moins 8 Go de RAM et Ubuntu Linux 24.04 installé dans une **partition dédiée d'au moins 30 Go**.
- Les distributions autres que Ubuntu Linux 24.04 ne sont pas supportées, et l'utilisation de Linux dans une machine virtuelle n'est également pas supportée.
- **Connexion à Internet rapide et sans filtrage** : au moins 50 Mbit/s de bande passante en téléchargement, et pas de filtrage des sites Web et protocoles.
- **Les ordinateurs contenant des données importantes doivent être sauvegardés** avant d'être utilisés dans nos sessions.

Modalités d'évaluation

Seuls les participants qui auront assisté à l'intégralité des journées de formation, et qui auront obtenu plus de 50% de réponses correctes à l'évaluation finale recevront une attestation individuelle de formation de la part de Bootlin.



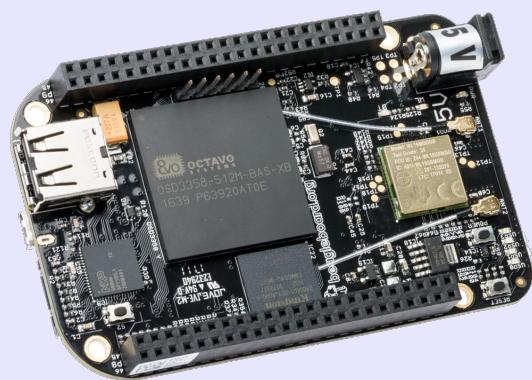
Handicap

Les participants en situation de handicap qui ont des besoins spécifiques sont invités à nous contacter à l'adresse training@bootlin.com afin de discuter des adaptations nécessaires à la formation.

Matériel

La plateforme matérielle utilisée pendant les travaux pratiques de cette formation est la carte **BeagleBone Black**, dont voici les caractéristiques :

- Un processeur ARM AM335x de Texas Instruments (à base de Cortex-A8), avec accélération 3D, etc.
- 512 Mo de RAM
- 2 Go de stockage eMMC embarqué sur la carte (4 Go avec la révision C)
- USB hôte et device
- Sortie HDMI
- Connecteurs à 2 x 46 broches, pour accéder aux UARTs, aux bus SPI, aux bus I2C, et à d'autres entrées/sorties du processeur.



Démonstrations

Les démos de cette formation utiliseront les périphériques matériels suivants :

- Une webcam USB
- Une carte d'extension d'écran tactile LCD connectée à la carte BeagleBone Black, pour afficher la vidéo capturée par la webcam.



1^{er} jour - Matin

Cours - Méthodes	TP - Construction du système
<ul style="list-style-type: none">• Comment mesurer le temps de démarrage• Principales approches	<ul style="list-style-type: none">• Téléchargement du code source du chargeur de démarrage, du noyau et de Buildroot• Prise en main de la carte, mise en place de la communication série• Configuration de Buildroot et génération du système• Configuration et compilation du chargeur de démarrage U-Boot. Préparation d'une carte SD pour démarrer le système.• Configuration et compilation du noyau. Démarrage du système.

1^{er} Jour - Après-midi

Cours - Mesure du temps	TP - Mesure du temps - Solution logicielle
<ul style="list-style-type: none">• Techniques génériques par logiciel• Techniques matérielles• Solutions spécifiques à chaque étage du démarrage	<ul style="list-style-type: none">• Modification du système pour mesurer le temps au niveau des différentes étapes.• Chronométrer les messages sur la console série• Chronométrer le démarrage de l'application

Cours - Optimisations des chaînes de compilation
<ul style="list-style-type: none">• Introduction aux chaînes de compilation• Bibliothèques C• Informations de taille• Mesure de la performance d'un exécutable avec la commande <code>time</code>



TP - Optimisations des chaînes de compilation

- Mesure du temps d'exécution de l'application
- Passage à une chaîne Thumb2
- Génération d'un SDK Buildroot pour recompiler plus vite

2^{ème} Jour - Matin

Cours - Optimisation de l'application

- Utilisation de `strace` et `ltrace`
- Autres techniques de profiling

TP - Optimisation de l'application

- Rechercher d'options de configuration inutiles dans des applications
- Modification de ces options à travers Buildroot
- Expériences avec `strace` pour suivre l'exécution d'un programme

Cours - Optimisation du démarrage du système

- Utilisation de BusyBox `bootchartd`
- Optimisation des scripts d'init
- Possibilité de démarrer directement votre application

TP - Optimisation du démarrage du système

- Utilisation de Buildroot pour supprimer scripts et commandes non nécessaires
- Une méthode pour identifier tous les fichiers inutilisés
- Simplification de BusyBox
- Démarrage de l'application en tant que programme init.



2^{ème} Jour - Après-midi

Cours - Optimisations de systèmes de fichiers

- Systèmes de fichiers disponibles, aspects de performance et de temps de démarrage
- Comment accélerer UBIFS
- Paramètres pour réduire le temps de démarrage
- Démarrer depuis un initramfs
- Utilisation d'exécutables statiques : contraintes de licence

TP - Optimisations de systèmes de fichiers

- Essayer et mesurer deux systèmes de fichiers bloc : ext4 et SquashFS.
- Essai et benchmark de la solution initramfs. Contraintes en rapport avec cette solution.

Cours - Optimisations du noyau

- Utilisation d'*Initcall debug* to générer un *boot graph*
- Options de compression et liées à la taille
- Réduction ou suppression de la sortie console
- Plusieurs réglages pour réduire le temps de démarrage

TP - Optimisations du noyau

- Génération et analyse d'un *boot graph* pour le noyau
- Identifier et éliminer les fonctionnalités du noyau non nécessaires
- Trouver la meilleure option de compression pour votre système

3^{ème} Jour - Matin

TP - Optimisations du noyau

- Poursuite du TP



3^{ème} Jour - Après-midi

Cours - Optimisations du chargeur de démarrage

- Conseils génériques pour réduire la taille et le temps de démarrage d'U-Boot.
- Optimisation des scripts d'U-Boot et du chargement du noyau
- Sauter le chargeur de démarrage - Comment modifier U-Boot pour activer son *Falcon mode*

Cours - Le *Falcon mode* d'U-Boot

- Principes et objectifs
- Prétraitement effectué par U-Boot pour préparer le démarrage de Linux
- Utilisation de la commande `spl export` pour faire ce traitement à l'avance.
- Modification du code source d'U-Boot et configuration pour démarrer directement Linux et sauter le deuxième étage d'U-Boot.
- Exemples and instructions de mise en oeuvre sur MMC et flash NAND
- Comment débugger le Falcon mode
- Comment revenir à U-Boot
- Limitations

TP - Optimisations du chargeur de démarrage

- Utilisation des techniques ci-dessus pour rendre le chargeur de démarrage le plus rapide possible
- Passer à un stockage plus rapide
- Sauter le chargeur de démarrage avec le *Falcon mode* d'U-Boot

Conclusion - Résultats obtenus

- Partage et comparaison des résultats obtenus par les différents groupes
- Questions / réponses, partage d'expérience avec le formateur