

L'utilisation de Linux dans les environnements industriels, pour des applications à fortes contraintes temporelles ou sur des systèmes retréints nécessite une bonne connaissance des mécanismes sous-jacents, comme l'ordonnancement des tâches, la gestion mémoire, ou le chargement des drivers du noyau.

Ce cours vous propose une exploration en profondeur du système Linux, de ses possibilités - et de ses limites - pour les applications temps-réel et les systèmes embarqués.

Organisation

Durée : 4 jours.

Pré-requis : Connaissance de Linux (utilisateur) et bonne familiarité avec le langage C.

Conseil cursus : La formation Ecriture de Drivers pour Linux représente un bon complément à ce cours.

Thèmes abordés

- **Développement industriel sous Linux** : outils de développement et de mise au point des applications, utilisation de l'environnement Eclipse, compilation et débogage croisés pour une cible embarquée, ajustement et compilation du noyau Linux.
- **Linux embarqué** : construction d'un noyau et d'un environnement réduit pour une cible embarquée, étude du boot du système, compilation et exécution des applications personnalisées.
- **Développement multi-tâches sous Linux** : processus et threads, communication et synchronisation entre tâches, gestion de la mémoire.
- **Temps-réel sous Linux** : principe de l'ordonnancement sous Linux, temps-partagé et temps-réel souple sous Linux, solutions temps-réel strictes avec RTAI et Xenomai.

Plan détaillé

I - Linux en environnement industriel

Linux et les logiciels libres

Présentation des concepts, des principes et des pratiques. Projet Gnu. Noyaux et distributions Linux.

Licences libres

Principes des GPL, LGPL, BSD... et implications pour le développement industriel.

Outils de développement libres

Chaîne de compilation Gnu, outils de débogage et de mise au point.

Eclipse et le CDT

Environnement de développement intégré. Création de projet, compilation et débogage.

Travaux pratiques

Utilisation du compilateur GCC, effets des différentes options. Détection d'erreurs à la compilation, à l'édition des liens. Débogage en cours de fonctionnement avec GDB. Débogage post-mortem avec GDB. Statistiques d'exécution et tests en couverture. Création et compilation d'application avec Eclipse. Utilisation du débogueur intégré.

.../...

II - Linux embarqué - Noyau

Linux sur cible embarquée

Plateformes de développement et d'exécution, type de cibles, utilisation d'un émulateur.

Chaîne de compilation croisée

Principe, création et mise en oeuvre d'une chaîne de compilation croisée. Utilisation de *Buildroot*. Environnement Eclipse pour la compilation croisée.

Compilation du noyau Linux.

Choix d'une version. Configuration et compilation.

Installation sur cible

Transfert de l'image du noyau. Configuration du *bootloader* ou de l'émulateur.

Système de fichiers

Types de système de fichiers, choix. Formatage et création de l'arborescence. Fichiers spéciaux des périphériques.

Travaux pratiques

Utilisation d'une chaîne de compilation pour *PowerPC*. Création d'une chaîne de compilation pour processeur *Arm*. Compilation et installation d'un noyau Linux pour cible *Arm*. Préparation d'un système de fichiers minimal.

III - Linux embarqué - Applications

Utilitaires système

Fonctionnement du processus *init*. Scripts de démarrage. Compilation de *Busybox*.

Bibliothèques et éditions des liens

Choix des bibliothèques nécessaires. Compilation de bibliothèques statiques ou dynamiques.

Débogage et mise au point

Débogage distant avec GDB et Eclipse. *Profiling* et tests en couverture.

Travaux pratiques

Compilation d'utilitaires système avec *Busybox*. Personnalisation des scripts de démarrage. Débogage et optimisation d'applications. Création de bibliothèques diverses.

IV - Multi-tâches sous Linux

Processus et threads

Création, exécution, terminaison, attente d'une autre tâche.

Communications entre processus

Files de messages Posix et segments de mémoire partagée.

Synchronisation et notification

Mutex, sémaphores Posix et signaux Unix.

Éléments temporels

Obtenir l'heure, précision, timers.

Gestion de la mémoire

Allocation et libération dynamiques, fiabilité et débogage.

Travaux pratiques

Création de processus et attente de processus fils. Passage à l'état Zombie. Création de communication entre threads. Utilisation des IPC Posix (mémoire partagée, sémaphores et files de messages). Test de saturation de la mémoire, désactivation de l'*overcommit*.

.../..

V - Temps-partagé et temps-réel souple sous Linux

Temps-partagé

Principe, configuration, efficacité, préemptibilité du noyau.

Temps-réel souple (*Soft Realtime*)

Principe, priorités, ordonnancements RR et FIFO.

Configuration du temps-réel

Passage en temps-réel, spécificité des noyaux postérieurs au 2.6.21.

Problèmes temps-réel classiques

Synchronisation des démarrages, inversion de priorité...

Limites du temps-réel souple

Fluctuation des timers, traitement des interruptions.

Travaux pratiques

Influences des priorités temps-partagé. Passage en temps-réel. Effets des boucles actives en temps-réel. Mesure de variation des timers.

VI - Temps-réel strict avec Linux

Principes du temps-réel strict (*Hard Realtime*)

Noyau standard et extensions *RT-Linux*, *RTAI*, *Xenomai*...

Installation de *Xenomai* ou *RTAI*

Téléchargement et application des *patches*, compilation et chargement.

Utilisation de *Xenomai*

Survol des API native et Posix de *Xenomai*

Travaux pratiques

Compilation et installation de *Xenomai*. Test de précision des timers.

Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

Travaux pratiques

Expérimentations libres suivant les demandes des participants.