D1S - Linux embarqué avec Ac6 System Workbench

Mettre en œuvre Linux sur les systèmes embarqués

Objectifs

- Comprendre l'architecture du système Linux
- Apprendre à installer Linux sur votre matériel et à créer un BSP
- Explorer l'architecture du système Linux
- Démarrer Linux
- Initialiser le système
- Installer des paquets existants sur la cible
- Apprendre à installer Linux sur des mémoires flash

Environnement du cours

- Support de cours imprimé (en anglais)
- Un PC Linux pour deux stagiaires.
- Une plateforme cible pour deux stagiaires

Prérequis

- Bonne maîtrise de la programmation C
- Connaissance de la programmation utilisateur sous Linux (voir notre cours D0 - Programmation en mode utilisateur Linux)
- De préférence, connaissance du kernel Linux et de la programmation de drivers (voir notre cours D3 - Drivers Linux)

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus.

Plan du cours

Premier jour

Introduction à Linux

- Historique de Linux et gestion des versions
- Architecture du système Linux
  - Processus et MMU
  - Appels système
  - Bibliothèques partagées
- Les composants de Linux
  - Toolchain
  - Bootloader
  - Kernel
  - Système de fichiers racine
- Les distributions Linux
- Les paquets Linux
- Les différentes licences utilisées par Linux (GPL, LGPL, etc)

Les outils Linux pour les systèmes embarqués

- Les bootloaders (UBoot, Redboot, barebox)
- Les bibliothèques optimisées (eglibc, uClibc)
- Les toolchains
- Les interfaces graphiques embarquées
- Busybox
- Les distributions embarquées
  - Commerciales
  - Standard
  - Outils pour construire des distributions sur mesure

Introduction à System Workbench

- Présentation
  - Eclipse
  - Kernel et modules
  - Plateformes et systèmes de fichiers racine
- L'architecture du système de build
  - Construire des paquets individuels
  - Construire des plateformes
  - Construire des systèmes de fichiers racine

Développer des applications avec System Workbench

- Créer un programme Linux
- Créer une bibliothèque
  - Bibliothèque statique
  - Bibliothèque partagée
- Déboguer sur la cible
  - Utiliser une connexion SSH
  - Déboguer des bibliothèques partagées

Utiliser U-Boot

- Introduction à U-Boot
- Démarrer la carte avec U-Boot
  - Démarrer depuis la NOR
  - Démarrer depuis la NAND
  - Démarrer depuis l'eMMC
- Les variables d'environnement d'U-Boot
  - Variables définies par l'utilisateur
  - Variables prédéfinies
  - Substitution de variables
- Le shell minimal d'U-Boot
  - Écrire des scripts dans des variables
  - Exécuter des scripts
  - Utiliser des variables dans les scripts : le motif set-script
- Les principales commandes d'U-Boot
  - Démarrer un OS
  - Accéder aux mémoires flash
  - Accéder aux systèmes de fichiers (NFS, FAT, EXTx, JFFS2&)
- Le shell complet d'U-Boot
  - Structure d'un script
  - Instructions de contrôle de flux (if, for&)
- Démarrer Linux
  - Les paramètres du kernel Linux

- La séquence de démarrage de Linux

Deuxième jour

Construire U-Boot

- Construire et installer U-Boot avec son système de build natif
- Construire U-boot avec System Workbench

Construire le kernel

- Le système de build de Linux
  - Télécharger le code source stable
- Récupérer une archive tarball
- Utiliser GIT
  - Configurer le kernel
  - Compiler le kernel et ses modules
- Modules livrés dans l'arbre des sources
- Modules hors arbre des sources
  - Installer le kernel et les modules
- Projets kernel
  - Créer un projet kernel
  - Choisir l'architecture et la configuration
  - Personnaliser la configuration
  - Compiler le kernel
- Projets de modules
  - Créer un projet de module
  - Le relier à un projet kernel
  - Créer et construire des modules

Construire des paquets

- Les paquets
  - Outils pour construire des paquets (gcc, Makefile, pkg-config&)
  - Autotools
  - Compiler de façon croisée un paquet avec les autotools
- Les applications tout-en-un
  - Busybox, les utilitaires de base
  - Dropbear : communications chiffrées (ssh)
- Démarrer automatiquement un programme au boot
  - Les systèmes d'initialisation (busybox init, system V init)

Créer une plateforme Linux

- Créer un projet de plateforme
  - Importer une plateforme préconfigurée
  - Créer une plateforme de zéro
- Configurer la plateforme
  - Répertoires de sources et d'installation
  - Lien vers un rootfs cible
  - Configurations de build
- Peupler l'environnement de build
  - Importer des paquets dans l'environnement de build
  - Construire des paquets individuels
  - Construire la plateforme complète
- Ajouter des paquets à une plateforme
  - Depuis une bibliothèque

    - Depuis un projet Eclipse existant
- Créer un nouveau paquet
    - Spécifier les sources
    - Patcher les sources officielles
    - Ajouter des ressources propres au paquet
    - Ajouter des directives de configuration du paquet

Troisième jour

Les systèmes de fichiers embarqués

- Les interfaces de stockage
    - Périphériques bloc
    - MTD
- Les mémoires flash et les MTD de Linux
    - Flashs NOR
    - Flashs NAND
    - Flashs ONENAND
- Les différents formats de systèmes de fichiers pour flash
    - JFFS2, YAFFS2, UBIFS
- Systèmes de fichiers en lecture seule
    - CRAMFS, SQUASHFS
- Les systèmes de fichiers Linux standard
    - Ext2/3/4, FAT, NFS
    - Ramdisks et initrd
    - Créer un initramfs
    - Démarrer via un initramfs
- Choisir les bons formats de système de fichiers
- Flasher le système de fichiers

Créer un système de fichiers racine

- Construire manuellement votre système de fichiers racine
    - Noeuds de périphériques, programmes et bibliothèques
    - Fichiers de configuration (réseau, udev, &)
    - Installer les modules
    - Rechercher et installer les bibliothèques nécessaires
    - Tester la cohérence et la complétude du système de fichiers
- La gestion de paquets
    - ipkg
- Créer un projet rootfs
    - Créer la structure du rootfs
    - Ajouter des fichiers à la structure de base
- Éditer les fichiers de configuration standard
    - Systèmes de fichiers
    - Initialisation
    - Lancement des applications