

Szkolenie: The Linux Foundation LFD435 Developing Embedded Linux Device Drivers



DOSTĘPNE TERMINY

2021-04-26 | 4 dni | Virtual Classroom
2021-04-26 | 4 dni | Wirtualna sala
2021-05-24 | 4 dni | Virtual Classroom

Cel szkolenia:

Ten 4-dniowy kurs ma na celu pokazanie doświadczonym programistom, jak tworzyć sterowniki urządzeń dla wbudowanych systemów Linux, i dać im podstawowe zrozumienie oraz znajomość jądra Linux.

Po opanowaniu tego materiału będziesz zaznajomiony z różnymi rodzajami sterowników urządzeń używanych w Linux i zostaniesz wprowadzony do wielu odpowiednich interfejsów API, które będą używane podczas pisania sterowników urządzenia.

Podczas gdy omówimy wewnętrzne elementy jądra i algorytmy, przeanalizujemy głęboko tylko te funkcje, które są zwykle używane w sterownikach urządzeń. Więcej szczegółów na temat takich rzeczy, jak planowanie, zarządzanie pamięcią, itp., odnaleźć można w bardziej skoncentrowanym na jądrze kursie.

Plan szkolenia:

- Wprowadzenie
 - Przedmiot szkolenia
 - Kim jesteś
 - The Linux Foundation
 - Szkolenia The Linux Foundation
 - Programy certyfikacyjne i cyfrowe identyfikatory
 - Dystrybucje Linux
 - Przygotowanie systemu
 - Rzeczy zmieniają się w Linux
 - Dokumentacja i linki
- Czynności przygotowawcze
 - Procedury

- Wersje jądra
- Źródła jądra i użycie Git
- Sprzęt
- Jak pracować w projektach OSS **
 - Omówienie wniesienia prawidłowego udziału
 - Badanie i zrozumienie DNA projektu
 - Określ czym chcesz się zająć
 - Zidentyfikowanie osób odpowiedzialnych, ich przepływu pracy i metod
 - Uzyskaj wczesne wejście i pracę w otwartym środowisku
 - Przekazuj przyrostowe bity, zamiast dużych fragmentów kodu
 - Zostaw swoje ego przed drzwiami: nie bądź wrażliwy
 - Bądź cierpliwy, rozwijaj relacje długoterminowe, bądź pomocny
- Cross-Development Toolchain
 - The Compiler Triplet
 - Wbudowany Linux Distribution Cross Compiler
 - Linaro
 - CodeSourcery
 - Crosstool-ng
 - Buildroot
 - OpenEmbedded
 - Yocto
- QEMU
 - Czym jest QEMU?
 - Emulowane architektury
 - Formaty obrazu
 - Dlaczego używać QEMU?
 - Laboratorium
- Podstawowa konfiguracja Target Development Board z uSD
 - Dlaczego używamy kart uSD?
 - Uzyskanie SW na karcie uSD
 - Dlaczego używanie kart uSD jest złym pomysłem?
 - Laboratorium
- Uruchamianie Target Development Board przez Ethernet
 - Korzystanie ze sprzętu zwirtualizowanego
 - Łatwiejszy sposób na rozwój
 - Cele laboratorium

- Laboratorium
- Konfiguracja jądra, kompilacja, uruchamianie
 - Konfiguracja jądra dla Development Board
 - Laboratorium
- Sterowniki urządzeń
 - Typy urządzeń
 - Mechanizm a polityka
 - Unikanie bloków binarnych
 - Zarządzanie energią
 - Jak aplikacje korzystają ze sterowników urządzeń
 - Przechodzenie przez wywołania systemowe dostępu do urządzenia
 - Numery błędów
 - Printk()
 - Devres: Managed Device Resources
 - Laboratorium
- Moduły i sterowniki urządzeń
 - Makra module_driver()
 - Moduły i Hot Plug
 - Laboratorium
- Zarządzanie pamięcią i alokacją
 - Pamięć wirtualna i fizyczna
 - Strefy pamięci
 - Tabela stron
 - Kmalloc()
 - _get_free_pages()
 - Vmalloc()
 - Płyty i alokacje pamięci podręcznej
 - Laboratorium
- Urządzenia znakowe
 - Węzły urządzeń
 - Liczby główne i drugorzędne
 - Rezerwacja dużych/mniejszych numerów
 - Dostęp do węzła urządzenia
 - Rejestracja urządzenia
 - Udev
 - Dev_printk() i Associates

- Struktura file_operations
- Punkty wejścia sterowników
- Struktury plików i Inode
- Różne sterowniki znaków
- Laboratorium
- Funkcje jądra
 - Komponenty jądra
 - Przestrzeń użytkownika a przestrzeń jądra
 - Czy są wywołania systemowe?
 - Dostępne wywołania systemowe
 - Algorytmy planowania i struktury zadań
 - Kontekst procesu
 - Laboratorium
- Przesyłanie między użytkownikiem a przestrzenią jądra
 - Przesyłanie między przestrzeniami
 - Put(get)_user() i copy_to(from)_user()
 - Bezpośredni transfer: Jądro I/O i mapowanie pamięci
 - Jądro I/O
 - Mapowanie stron użytkownika
 - Mapowanie pamięci
 - Funkcje przestrzeni użytkownika dla mmap()
 - Punkt wejścia sterownika dla mmap()
 - Dostęp do plików z jądra
 - Laboratorium
- Sterowniki platformy
 - Czym są sterowniki platformy?
 - Główne struktury danych
 - Rejestrowanie urządzeń platformy
 - Przykład
 - Twarde dane platformy
 - Nowa droga: drzewa urządzeń
 - Laboratorium
- Drzewa urządzeń
 - Czym są drzewa urządzeń?
 - Co robią a czego nie drzewa urządzeń
 - Składnia drzewa urządzeń

- Przechodzenie przez drzewa urządzeń
- Powiązania drzewa urządzeń
- Obsługa drzewa urządzeń w programach rozruchowych
- Korzystanie z danych drzewa urządzeń w sterownikach
- Współistnienie i konwersja starych sterowników
- Laboratorium
- Przerwania i wątki
 - Czym są przerwania i wątki?
 - Wyjątki
 - Przerwania asynchroniczne
 - MSI
 - Aktywowanie/dezaktywowanie przerw
 - Czego nie można robić w trakcie przerwania
 - Struktura danych IRQ
 - Instalowanie programu obsługi przerw
 - Laboratorium
- Pomiary czasu
 - Rodzaje pomiarów czasu
 - Jiffies
 - Uzyskiwanie aktualnego czasu
 - Źródła zegara
 - Zegar czasu rzeczywistego
 - Programowalny interwalometr
 - Licznik znaczników czasu
 - HPET
 - Going Tickless
- Zegary jądra
 - Wstawianie opóźnień
 - Czym są zegary jądra?
 - Funkcje zegara o niskiej rozdzielczości
 - Implementacja zegara o niskiej rozdzielczości
 - Zegary wysokiej rozdzielczości
 - Korzystanie z zegarów wysokiej rozdzielczości
 - Laboratorium
- loctls
 - Czym są loctls?

- Punkty wejścia sterowników dla loctls
- Zablokowane i bezzamkowe loctle
- Definiowanie loctls
- Laboratorium
- Unified Device Model i sysfs
 - Unified Device Model
 - Podstawowe struktury
 - Prawdziwe urządzenia
 - Sysfs
 - Przykłady kset i kobject
 - Laboratorium
- Firmware
 - Czym jest Firmware?
 - Ładowanie Firmware
 - Laboratorium
- Usypianie i Kolejki Oczekujące
 - Czym są Kolejki Oczekujące?
 - Usypianie i wybudzanie
 - Przechodzenie do szczegółów usypiania
 - Ekskluzywne usypianie
 - Szczegóły wybudzania
 - Ankietowanie
 - Laboratorium
- Obsługa przerwania: Odroczone funkcje i sterowniki użytkownika
 - Połówki górne i dolne
 - Softirqs
 - Zadania
 - Kolejki Robocze
 - Nowe API Kolejki Roboczej
 - Tworzenie wątków jądra
 - Wątkowe programy obsługi przerwania
 - Obsługa przerwania w przestrzeni użytkownika
 - Laboratorium
- Sprzęt I/O
 - Bariery pamięci
 - Przydzielanie i mapowanie pamięci I/O

- Dostęp do pamięci I/O
- Bezpośredni dostęp do pamięci (DMA) **
 - Czym jest DMA?
 - Ograniczenia pamięci DMA
 - Maski DMA
 - DMA API
 - Pule DMA
 - Scatter/Gather Mappings
 - Laboratorium
- Urządzenia technologii pamięci - MTD (System Plików Pamięci Flash)
 - Czym są urządzenia MTD?
 - NAND vs. NOR vs. eMMC
 - Moduły sterownika i użytkownika
 - System plików Flash
 - Laboratorium
- Sterowniki USB
 - Czy jest USB?
 - Topologia USB
 - Terminologia
 - Punkty końcowe
 - Deskryptory
 - Klasy urządzeń USB
 - Obsługa USB w Linux
 - Rejestrowanie sterowników urządzeń USB
 - Przenoszenie danych
 - Przykład sterownika USB
 - Laboratorium
- Ankieta zamknięcia i oceny
 - Ankieta ewaluacyjna

** Te sekcje mogą być uznane za częściowo lub w całości opcjonalne. Zawierają materiały źródłowe, tematy specjalistyczne lub przedmioty zaawansowane. Instruktor może zdecydować się na ich pokrycie lub nie, w zależności od doświadczenia uczestników i ograniczeń czasowych.

Wymagania:

Aby jak najlepiej wykorzystać kurs powinieneś posiadać:

Znajomość podstawowych interfejsów i metod jądra, takich jak pisanie, kompilowanie, ładowanie i rozładowywanie modułów, używanie podstawowych operacji synchronizacji oraz podstaw alokacji i zarządzania pamięcią, takich jak LFD420 (Kernel Internals and Development).

Poziom trudności



Certyfikaty:

Uczestnicy otrzymają certyfikaty podpisane przez The Linux Foundation.

Prowadzący:

Certyfikowany trener The Linux Foundation.