

## Szkolenie: The Linux Foundation LFD420 Linux Kernel Internals and Development?



### DOSTĘPNE TERMINY

2021-05-10 | 4 dni | Virtual Classroom

2021-05-10 | 4 dni | Wirtualna sala

### Cel szkolenia:

Ten czterodniowy kurs ma na celu zapewnienie doświadczonym programistom solidnego zrozumienia jądra Linuxa. Oprócz szczegółowego spojrzenia na teorię i filozofię jądra Linuxa, będziesz także uczestniczył w rozległych ćwiczeniach praktycznych i demonstracjach zaprojektowanych w celu dostarczenia niezbędnych narzędzi do rozwijania i debugowania kodu jądra Linuxa.

Na tym kursie dowiesz się między innymi:

- Jak zaprojektowano Linuxa
- Jak działają algorytmy jądra
- Jak zarządzać sprzętem i pamięcią
- O technikach modularyzacji i debugowania
- Jak działa społeczność programistów jądra i jak skutecznie z nią pracować

Informacje zawarte w tym kursie będą działać z każdą główną dystrybucją Linuxa.

### Plan szkolenia:

- Wprowadzenie
  - Cele
  - Przedstawienie uczestników
  - The Linux Foundation
  - Linux Foundation Training
  - Dystrybucje Linux
  - Platformy
  - Przygotowanie twojego systemu
  - Pobranie i używanie wirtualnej maszyny
  - Rzeczy zmieniające się w Linux
  - Dokumentacja i linki

- Rejestracja kursu
- Czynności wstępne<
  - Procedury
  - Wersje jądra
  - Źródła jądra i zasotosowanie git
- Jak pracować w projekcie Open Source \*\*
  - Omówienie jak prawidłowo współpracować w takim projekcie
  - Skup się na bezpieczeństwie i jakości
  - Badanie i zrozumienie DNA projektu
  - Dowiedz się nad czym chcesz pracować
  - Identyfikacja opiekunów i ich przepływy pracy i metody
  - Uzyskanie wczesnego wejście i pracy w otwartym środowisku
  - Przekazuj przyrostowe bity, a nie duże zrzuty kodu
  - Zostaw swoje ego za drzwiami
  - Bądź cierpliwy, rozwijaj relacje długoterminowe, bądź pomocny
- Architektura jądra I
  - UNIX i Linux \*\*
  - Monolityczne i mikro jądra
  - Metody obiektowe
  - Główne zadania jądra
  - Przestrzeń użytkownika i przestrzeń jądra
  - Linux w trybie jądra
- Ogólny zarys programowania jądra
  - Numery błędów i pobieranie wyjścia jądra
  - Struktura zadań
  - Przydziały pamięci
  - Przesyłanie danych między użytkownikiem a przestrzeniami jądra
  - Połączone listy
  - Konwersja String'ów na liczby
  - Jiffies
  - Laboratoria
- Moduły
  - Czym są moduły?
  - Trywialny przykład
  - Kompilacja modułów
  - Moduły vs Built-in

- Narzędzia do obsługi modułów
- Automatyczne ładowanie / rozładowywanie modułów
- Licznik użycia modułu
- Struktura modułu
- Licencjonowanie modułu
- Eksportowanie symboli
- Rozwiązywanie symboli \*\*
  - Laboratoria
- Architektura jądra II
  - Procesy, wątki i zadania
  - Kontekst proces
  - Łątka na jądro w czasie rzeczywistym
  - Dynamiczne łatanie jądra
  - Alternatywy wdrażania\*\*
  - Migracja do nowej platformy \*\*
  - Laboratoria
- Inicjalizacja jądra
  - Ogólny zarys inicjalizacji jądra
  - Rozruch systemu
  - Das U-Boot dla systemów wbudowanych\*\*
- Konfiguracja i kompilacja jądra
  - Instalacja i układ źródła jądra
  - Przeglądarki jądra
  - Pliki konfiguracyjne jądra
  - Budowanie jądra i pliki Makefile
  - Initrd i initramfs
  - Laboratoria
- Połączenia systemowe
  - Czym są połączenia systemowe?
  - Dostępne połączenia systemowe
  - W jaki sposób realizowane są wywołania systemowe
  - Dodanie nowego wywołania systemowego
  - Laboratoria
- Styl jądra i uwagi ogólne
  - Styl kodowania
  - kernel-doc \*\*

- Używanie ogólnych procedur i metod jądra
- Tworzenie poprawki jądra
- sparse
- Używanie likely() i unlikely()
- Pisanie przenośnego kodu, CPU, 32/64-bit, Kolejność bajtów
- Pisanie dla SMP (Symetric Multiprocessing)
- Pisanie dla HMS (High Memory Systems)
- Zarządzanie zasilaniem
- Dbanie o bezpieczeństwo
- Mieszanie nagłówków przestrzeni użytkownika i jądra\*\*
- Laboratoria
- Metody synchronizacji
  - Współbieżność i metody synchronizacji
  - Operacje atomowe
  - Operacje bitowe
  - Spinlocks
  - Seqlocks
  - Wyłączenie przywłaszczenia
  - Muteksy
  - Semafony
  - Funkcje zakończenia
  - Read-Copy-Update (RCU)
  - Liczniki referencyjne
  - Laboratoria
- SMP i wątki
  - Jądra i moduły SMP
  - Koligacja procesora
  - CPUSETS
  - Algorytmy SMP - planowanie, blokowanie itp.
  - Zmienne na procesor \*\*
  - Laboratoria
- Procesy
  - Czym są procesy?
  - task\_struct
  - Tworzenie procesów użytkownika i wątków
  - Tworzenie wątków jądra

- Niszczenie procesów i wątków
- Wykonywanie procesów przestrzeni użytkownika z jądra
- Laboratoria
- Ograniczenia i możliwości procesu \*\*
  - Limity procesowe
  - Możliwości
  - Laboratoria
- Monitorowanie i debugowanie
  - Pakiety Debuginfo
  - Śledzenie i profilowanie
  - sysctl
  - Klucz SysRq
  - Wiadomości oops
  - Debugery jądra
  - debugfs
  - Laboratoria
- Planowanie
  - Główne zadania planowania
  - SMP
  - Priorytety planowania
  - Planowanie połączeń systemowych
  - Funkcja 2.4 schedule ()
  - Harmonogram O(1)
  - Odstępy czasu i priorytety
  - Równoważenie obciążenia
  - Priorytetowa inwersja i dziedziczenie priorytetowe \*\*
  - Harmonogram CFS
  - Obliczanie priorytetów i sprawiedliwych czasów
  - Planowanie zajęć
  - Szczegóły harmonogramu CFS
  - Laboratoria
- Adresowanie pamięci
  - Zarządzanie pamięcią wirtualną
  - Systemy z i bez MMU i TLB
  - Adresy pamięci
  - Wysoka i niska pamięć

- Strefy pamięci
- Specjalne węzły urządzeń
- NUMA
- Stronicowanie
- Tabele stron
- struktura strony
- Kernel Samepage Merging (KSM) \*\*
- Laboratoria
- Ogromne strony
  - Obsługa ogromnych stron
  - libhugetlbfs
  - Przezroczyste ogromne strony
  - Laboratoria
- Przydział pamięci
  - Żądanie i zwalnianie stron
  - Buddy System
  - Płyty i alokacje pamięci podręcznej
  - Pule pamięci
  - kmalloc()
  - vmalloc()
  - Wczesne alokacje i bootmem()
  - Defragmentacja pamięci
  - Laboratoria
- Przetwarzanie przestrzeni adresowej
  - Przydzielanie pamięci użytkownika i przestrzeni adresowej
  - Blokowanie stron
  - Deskryptory pamięci i regiony
  - Prawa dostępu
  - Przydzielanie i zwalnianie regionów pamięci
  - Błędy stron
  - Lab
- Cache dyskowy i swapping
  - Czym jest cache?
  - Podstawy cache'a stron
  - Czym jest Swapping?
  - Obszary wymiany

- Zamiana stron na zewnątrz i na zewnątrz
- Kontrolowanie Swappiness
- Pamięć podręczna wymiany
- Odwrotne mapowanie \*\*
- OOM Killer
- Laboratoria
- Sterowniki urządzeń\*\*
  - Typy urządzeń
  - Węzły urządzeń
  - Sterowniki znaków
  - Przykład
  - Laboratoria
- Sygnały
  - Czym są sygnały?
  - Dostępne sygnały
  - Wywołania systemowe dla sygnałów
  - Sigaction
  - Sygnały i wątki
  - W jaki sposób jądro obsługuje sygnały
  - Jak jądro wysyła sygnały
  - Jak jądro wywołuje procedury obsługi sygnałów
  - Sygnały w czasie rzeczywistym
  - Laboratoria
- Zakończenie i ankieta oceniająca

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

## Wymagania:

Uczniowie powinni biegle posługiwać się językiem programowania C, podstawowymi narzędziami Linux (UNIX), takimi jak ls, grep i tar, i czuć się swobodnie z dowolnym dostępnym edytorem tekstu (np. Emacs, vi itp.). Doświadczenie z jakąkolwiek dystrybucją Linuxa jest pomocne, ale nie jest wymagane.

## Poziom trudności



## Certyfikaty:

Uczestnicy otrzymają certyfikaty podpisane przez The Linux Foundation.

## Prowadzący:

Certyfikowany trener The Linux Foundation.