INTUITIVE EMBEDDED TECH

Interactive Tech Education

www.intuitiveembedded.com

Call: 9900721303 / 9892616426 | Email: info@intuitiveembedded.com

Complete Course on OS Concepts and Linux Programming

Total Duration for Complete Course:-
60 Hours (Only On SATURDAY and SUNDAY)

NOTE: Practice will be done on Linux (Ubuntu)

Why 'Linux system programming:-
Anybody who want to make a career in Linux or Embedded System should take this course. This Course covers all the OS concepts from basic to advance level and help to understand complete architecture of any Operating system.

The participant will develop a deep understanding of Linux or UNIX systems and learn concepts and skills which are essential for programming and software development on Linux-based platforms for both enterprise and embedded products and/or applications. The course is an in-depth coverage on Linux system fundamentals.

In kernel Programming section Participants will learn how Linux-C programming in kernel space is different than user space and they will be able to do their own kernel configuration and do kernel building from kernel source code. Subsequently, the course goes over various kernel sub-systems such as Kernel Virtual Memory, Process Management, Linux Scheduler, Kernel Synchronization Primitives, Kernel Time-keeping Architecture, Kernel Memory Management, Process Address Space, System Calls infrastructure in the Kernel, Signals, Virtual Filesystems, Page/Buffer Cache and Swapping in the Kernel.

Both Linux and Unix System have very similar programming environment. Application programs written on Unix will work on Linux systems with minimal or no changes. So, by undergoing this training program, one can master the programming skills both for Linux Systems as well as other Unix Systems like Solaris, HPUX, AIX, IRIX and other variants of Unix.

OS Topics

1. OS Concept
1. Computer system structures and Overview
2. Operating system structures and Overview
3. Process concept in modern operating systems
    1. User-Space and System-Space
    2. Process State, State Diagram and Process Address Space
    3. System Call and CPU Scheduler
    4. Process Vs Program
    5. Process Scheduling Algorithm
        1. FCFS
        2. RR
        3. Priority Based
        4. Yielding
    6. Ideal processes

7. Nice priorities and nice value
8. Inter-Process Communication (IPC)
    1. Why IPCs
    2. Type of IPC Mechanisms
        1. Pipe
            1. Pipes & FIFOs
            2. Full Duplex Pipes
            3. Persistence of Pipes & FIFOs
            4. Pros and Cons of Pipes/FIFOs
            5. Limitation of Pipes/FIFOs
        2. Signals
            1. Why Signals?
            2. Signals vs. Interrupts
            3. Signals in UNIX
        3. System V IPCs
            1. Shared Memory
            2. Message Queues
            3. Semaphores
                1. Binary & Counting Semaphore
        4. Socket
9. Process synchronization
    1. Race Condition
    2. Deadlock
    3. Critical Section
    4. Semaphores
    5. Mutex
    6. Spinlocks
10. Context Switching

4. System Call Handler
5. Interrupt Handler
6. Thread concept and multi-threading
7. Daemons Process
    1. User Vs Daemon Process
    2. Characteristics of a Daemon
8. Memory Management
    1. Segment based
    2. Page based
    3. Virtual Memory Management
        1. Paging & Swapping
        2. Page Table
        3. Memory Mapping
        4. TLB and Cache Memory
        5. Cache colouring and cache coherence
        6. Demand Paging
        7. Page Stealing Algorithm (LRU)
        8. Thrashing
9. Linux File-System
    1. File Types
    2. File Tree & Types
    3. File Systems
    4. Super Block & Inode
    5. System Vs Function Calls
    6. System Call Sequence
    7. File descriptor table

Linux Topics

1. Introduction to Linux

1. Introduction to UNIX
2. Linux Layered Architecture
3. Bootup Sequence
4. Linux Installation Guide
5. Getting familiar with Linux Environment
    1. Basic Linux Bash Command
    2. Vim Editor Guide
    3. Installation of software and packages.
    4. File structures and Layout
6. RTOS vs. Linux

2. Linux User-Space Programming

1. Process Management
    1. Process creation/ Termination/ management calls
        1. fork
        2. vfork
        3. clone
        4. exit
        5. wait+
        6. exec+
        7. signal
        8. alarm
        9. sleep
        10. kill and many other calls.
    2. Getting Identities of process
        1. pid
        2. ppid
        3. uid
        4. gid
        5. euid
        6. egid
        7. pgid
        8. sid and setting couple of these for tasks
    3. Scheduling functions
        1. Setting scheduling policies
        2. Nice value
    4. POSIX Thread creation/ Termination/ Management calls
        1. pthread_create
        2. pthread_detach
        3. pthread_exit
        4. pthread_join
        5. pthread_kill and many other calls.
2. IPC and Synchronization
    1. pipe
    2. shared memory
    3. message queues
    4. semaphores
    5. Signals

1. Signal Related System Calls
2. Signal Generation and Delivery
3. Pending Signals
4. Action Performed Upon Delivery
5. Signals @ Multithreaded Application
6. Data Structures for Signal Handling
7. Catching the Signal

3. Daemons development
    1. Writing a Daemon Code
    2. Timers & Resource Limits
    3. Interval Timers
    4. High Resolution Timers
    5. System calls for Timers
    6. Resource Limits
    7. Hard Limit / Soft Limit
4. File Management
    1. File related System Calls
        1. creat
        2. open
        3. read
        4. write
        5. close
        6. lseek
        7. stat
        8. fstat
        9. access
        10. dup+
        11. link, unlink and locking files etc.
    2. File Control Operations
    3. File Locking
    4. fcntl() calls
5. Environment functions:
    1. getenv
    2. setenv

3. Understanding Linux Kernel-Space

1. Getting Started with Kernel
    1. The Process/Kernel Model
    2. Diagram of Linux subsystems (Application, Kernel, and Hardware Relation)
    3. What is Monolithic and Microkernel Kernel?
    4. What is Re-entrant Kernel and Pre-emptive Kernel?
    5. Pre-emptive Context Switch
    6. Linux Kernel Version
    7. Obtaining and Installing Kernel source
    8. Creating and using the Patches
    9. The Kernel Source Tree
    10. Building the Kernel
    11. Configuring the Kernel
    13. Spawning Multiple Build Jobs
    14. Installing the New Kernel
    15. Role of the kernel
    16. Kernel documentation
2. Processes in Linux Kernel
    1. Process

2. Lightweight Process
3. Threads & Thread Groups
4. Process Descriptions
5. The task_struct
6. Linux Process States
7. Thread Group Leader
8. thread_info Structure
9. Kernel Stack Structure
10. The Process List
11. Waiting Processes
12. Wait Queues
13. Awakening Processes
14. Process Creation
15. clone () / fork() / vfork()
16. Kernel Threads
17. Process 0 & Process 1
18. Destroying Processes
19. exit_group () / _exit()
20. Process Removal
21. Various context (process, kernel, interrupt)

3. Process Scheduling
    1. Various Scheduling Algorithm
    2. Process Pre-emption and Kernel Pre-emption
    3. Quantum Duration
    4. Scheduling Policies
    5. Scheduling of Conventional Processes
    6. Dynamic Priority
    7. Real-time Processes
    8. Real-time Process Scheduling
    9. Run queue Data Structure
    10. Runqueue Balancing in Multiprocessing Systems
    11. Scheduler-Related System Calls
    12. Load Balancing
    13. CPU Binding (Affinity)
4. System Calls
    1. System Call Number
    2. System Call Handler
    3. Service Routines
    4. Invoking a System Call
    5. Leaving a System Call
    6. Arguments to System Call
    7. Verifying the Arguments
    8. Accessing Process Address Space
    9. Exception Tables
    10. Fixing Address Exceptions
    11. Kernel Wrapper Routines
    12. System Call Context
    13. Current pointer
    14. Implementing System Calls
    15. Accessing the System Call from User-Space
5. Kernel Data Structures
    1. Kernel Linked List
    2. Add Element to the List
    3. Delete Element from the List

3. Move Element from the List
4. container_of
5. KFIFO

6. Interrupt Handling
    1. Understanding Interrupts and Interrupt Context
    2. Linux Interrupt handlers & Interrupt Service Routines (ISR)
    3. RETI Instruction
    4. /proc/interrupts
    5. Registering an Interrupt Handler
    6. Freeing an Interrupt Handler
    7. Interrupt priorities & Shared Handlers
    8. Top Halves versus Bottom Halves
    9. Need for deferred routines
    10. Various Deferred Routines
        1. Task Queues
        2. Softirqs
        3. Tasklets
        4. Kernel Timers
        5. Work Queues
        6. Softirqs Vs Tasklets Vs Work Queues
    11. IO-APIC and APIC concepts
    12. Shared interrupt handler
7. Kernel Synchronization
    1. Kernel Pre-emption
    2. When Synchronization is Necessary
    3. Critical Regions and Race Conditions
    4. Concurrency
    5. When Synchronization is Not Necessary
    6. Synchronization Primitives
        1. Per-CPU Variables
        2. Atomic Operations
        3. Optimization & Memory Barriers
        4. Spin Locks
        5. Read_Locks/Write_Locks (Reader-Writer Spin Locks)
        6. Seqlocks
        7. Read-Copy Update (RCU) Locks
        8. Semaphores
        9. Read/Write Semaphores
        10. Mutex
        11. Local Interrupt Disabling
        12. Disabling Deferrable Functions
        13. Preemption Disabling
    7. Semaphores Vs Mutex Vs Spin Locks
8. Timing Measurements in Linux
    1. Clocks and Timer Circuits
    2. Linux Timekeeping Architecture
    3. The jiffies and Hz Variable
    3. Updating System Statistics
    4. Software Timers
    5. Dynamic Timer List Structure
    6. Timer List Data-Structure
    7. Per-CPU Timer List
    8. Delay Functions
    9. udelay () and ndelay()

10. System Calls for POSIX Timers and Clocks

9. Memory Management
    1. Page Frame / Page
    2. Memory Zones
    3. Reserved Page Frames
    4. High Memory Page Frames
    5. Permanent and Temporary Kernel Mappings
    6. Dynamic Contigous Page Allocation
    7. Type Flag and Modifiers
    7. Slab Allocator and Buddy Allocator
    8. Object Caches
    9. Slab Cache Allocation
    10. Object Allocation
    11. kmalloc (), vmalloc(), kfree(), vfree(),
    12. kmalloc () Vs vmalloc()
    13. Non-contigous Memory Area Management
10. Memory Addressing
    1. Logical, Linear & Physical Addresses
    2. Translating a Logical Address
    3. Segmentation in Linux
    4. Paging in Linux
    5. Large Pages
    6. Physical Page Extension (PAE)
    7. Hardware Cache
    8. Cache Coherency
    9. Translation Lookaside Buffers (TLBs)
    10. Linux Paging Model
    11. Physical Memory Layout
    12. Kernel Physical Memory Variables
    13. Process Page Tables
    14. Kernel Page Tables
    15. TLB Management
11. Page Frame Reclamation
    1. Page Frame Reclaiming Algorithm
    2. Unreclaimable pages
    3. Swappable Pages
    4. Syncable Pages
    5. Discardable Pages
    6. Design of the Algorithm
    7. Reclaimation Tigger Points
    8. Low on Memory
    9. Periodic Reclaiming
    10. The Out-of-memory Killer
    11. Swapping Features
12. Linux Page Cache
    1. Page Cache Contents
    2. Page Cache Requirements
    3. The address_space Object
    4. Block Buffers and Page Cache
    5. The Buffer Head
    6. Buffer Pages
    7. Writing Dirty Pages to Disk
    8. bdflush & pdflush
    9. sync (), fsync() and fdatasync() calls

13. Process Address Space
    1. Kernel Address Space
    2. Process Address Space
    3. System Calls for Memory Region
    4. brk (), execve(), _exit(), fork()
    5. mmap (), mmap2(), munmap(),
    6. shmat (), shmdt()
    7. Page Faults
    8. Valid & Invalid Addresses
    9. The Memory Descriptor
    10. The mem_struct Structure
    11. Memory Regions
    12. Linear Address Intervals – do_mmap ()
    13. Page Fault Exception Handling
    14. Demand Paging
    15. Copy on Write (COW)
    16. Creating and Deleting Process Address Space
    17. Managing Heap Space
14. Linux File System
    1. Linux Virtual Filesystem
    2. VFS Role in File Operations
    3. VFS Supported Filesystem Classes
    4. Common File Model
    5. Process & VFS Object Interaction
    6. VFS Data Structures
    7. Super Block Object
    8. Inode Object
    9. File Object
    10. Dentry Object
    11. Processes and Files
    12. The fd array
    13. Constraints on a Process
    14. File System Types
    15. Filesystem Type Registration
    16. Filesystem Handling
    17. Mounting a Filesystem
    18. Pathname Lookup
    20. Reads & Writes
    21. Other File Operations
15. Other Topics
    1. /proc and /sys virtual file system
    2. /proc/interrupts
    3. /proc/devices
    4. /proc/kallsyms
    5. Use of debugfs
    6. dmesg

THANK YOU