

Module: 1/7

Module name: Linux Systems

Overview	This module is a kick-start course intended to get new programmers up and running with Linux OS. This course will start with basics of Linux and its features, then covers substantial commands that will be used by a user and programmer. You will be learning Shell scripting, followed by programming tools including make, gdb, VIM. This module then deals with compilation stages, coding guidelines and creating projects using makefiles.
Description	If you are not familiar with Linux as a user or a developer, this course is a prerequisite for other courses delivered at EMERTXE. You will get familiar with the command line interface, the shell, and the essential commands which you will using every now and then as a Linux user/programmer. You will get to know the power of Linux by learning redirection, pipes and how to combine multiple commands together to automate tasks. Following this, we teach you about various stages involved in compilation using GCC, and writing Makefiles. Finally you will learn how to modularize and organize "C" source and header files to create projects and libraries .
Objective	<ul style="list-style-type: none"> ● To get familiar with Linux Operating systems, and various commands, and VIM editor. ● You should be able to write shell scripts ● To understand the different stages involved in code compilation. ● You should be able to know how to create projects and automate the build using makefiles.
Platform	Linux
Delivery method	Instructor led, Assignments, Mini Project.
Course topics	Overview of Linux OS, Command Line Interface - Shell, Environment variables, shell commands, filtering commands, user accounts, remote login, redirection, pipes, Directory and File system structure, Visual editor (VIM), Shell scripting, Makefiles, code compilation stages, coding guidelines and creation of project and libraries.

Module: 2/7

Module name: Advanced C

Overview	An assignment filled intense Advanced C programming course, taken from Problem to Logic till Program, concluded by an apt project.
Description	<p>Lot of book tells about what is programming. Many also tell how to write a program. But very few tells you about how to translate logic into a program. Specifically, in this fast paced industry, when you don't have time to think to program, this course comes really handy. It builds on the basics of programming, smooth sailing through the advanced nitty-grittys of the language and how to use it to translate logic, the industry-way.</p> <p>Every class is backed by discussion and topic related assignments. Finally a project letting you apply most of the concepts learned throughout the course.</p>
Objective	<ul style="list-style-type: none"> ● To have a clear concept of C language. ● To obtain good quality and style in programming. ● To induce confidence in candidates.
Platform	Any (with specific mention to Linux)
Delivery method	Instructor led, Assignments, Mini Project.
Course topics	Basics, Operators, Conditionals, Arrays, Pointers, Structures, Unions, Bit-wise operations, Functions, Files, Preprocessor directives, Recursion, Creating & Building a project, Makefiles, Logic to program translation, Creating your own library, Introduction to Data Structures.

Module: 3/7

Module name: Micro controllers

Overview	A practical approach on Micro controller with basic Hardware concept and Embedded C programming. A Practical approach on Micro-controller based application development with Basic Hardware concepts, Peripheral Programming, Protocols concluded by Project.
Description	<p>A complete course module dealing with how to write a embedded C program for a Hardware with any micro controller on it. Planning on the software and hardware optimization in application design for ease in development.</p> <p>This module is dealt with practical issues normally faced in development phase and how to overcome them. Building an complete application with the help of gained knowledge throughout the course as a project.</p>
Objective	<ul style="list-style-type: none"> ● To provide a basic idea of hardware/electronics aspects of programming, which an embedded engineer requires. ● On completion of this module the candidate should be able to program any micro controller and design basic application with ease.
Platform	<p>Software / Tools: Any (with specific mention to Linux)</p> <p>Hardware: 8051 and PIC based Architecture</p>
Delivery method	Instructor led, Assignments, Mini Project.
Course topics	Introduction to embedded systems, Microprocessors vs Micro controllers, Each session with hardware related concept on which the candidate will be working on, GPIOs, Analog I/Os, Types of memories and its usage ,interfacing etc., Basic micro controller peripherals such as Timers, Counters etc, Interrupts and its sources, Basic communication protocols like UART, SPI, I2C etc.

Module: 4/7

Module name: Data structures and Logic analysis

Overview	<p>An assignment filled intense course taking from problem to logic till program, concluded by an apt project. A firm understanding of data structures provides a basis for writing more efficient code. The four hallmarks of good programming are proper design, clear coding, good organization of data, and correct algorithm selection.</p> <p>This course is intended to provide an understanding of data specification and abstraction, how that leads to the specification of data structures, and how data structures are implemented as late as possible, based on the specified structures.</p>
Description	<p>This is an Advanced Data structure course suitable for anyone who would like to pursue their career in the field of programming and become expert in the field independent of the platform which he or she would be working. While taking the course the candidate is expected to be clear with the functional aspects of the language and concentrate only on improving his programming skills.</p>
Objective	<ul style="list-style-type: none"> ● Review methods for problem solving and algorithm analysis. ● Develop an understanding of Abstract Data Types (ADT) and their implementation. ● Understand the importance of information hiding, data abstraction, and modular design. ● Recognize programming needs, the cost and benefits of each data structure in your toolbox, and how to select the right tool for the job.
Platform	Any (with specific mention to Linux)
Delivery method	Instructor led, Assignments, Workshops, Project.
Course topics	Introduction to Data Structures and Abstract Data Types, Stacks, Queues, Searching, Sorting, Lists, Trees, Algorithm Analysis.

Module: 5/7

Module name: Linux Internals

Overview	This module is intended to build a strong knowledge about operating systems. This guides the programmer to build high performance embedded applications using standard Linux APIs and interfaces.
Description	<p>Currently most of the embedded systems are build over an operating system due to performance requirements and the resource complexity. This adds responsibility to developers to understand the make use of the operating system capabilities to build a secure, high performance and crash free system.</p> <p>This course gives complete understanding of the Operating system concepts and Linux internals (Interfaces, API's and system calls). This module helps the audience to move to the next level of programming by considering other factors in the system. This module is industrial aligned and provides ample practical classes to provide good exposure to Linux programming.</p>
Objective	<ul style="list-style-type: none"> ● To gain strong knowledge of OS programming ● Proficiency on the Linux API's and system calls ● To get the knowledge of high performance and secure coding by using OS capabilities.
Platform	Linux
Delivery method	Instructor led, Assignments, Workshops, Project(optional).
Course topics	OS Basics - Process, CPU scheduling, Dead lock & starvation, priority, Components of Linux - Kernel structure, Shell basics, Linux file system - ext2 & ext3, POSIX Threads - Multi threaded programming, p-thread API's, Synchronization - Race condition & mutex, semaphores, condition variables, memory barriers, spinlock, IPC - pipe, FIFO, shared memory, System V-semaphores, Process management and memory management.

Module: 6/7

Module name: Embedded Linux on ARM 9

Overview	A course which prepares you on the most recent and changing domain of computer industry the embedded systems. This course will tell you everything starting from basic things like cross compilers etc and will cover all the embedded OS development related things like making custom platform kernels, boot loaders creating file systems setting up development environment. and all the theoretical background needed for this.
Description	<p>The course is unique in terms that it combines your skills of Linux administration, Hardware knowledge, Linux as OS, C and computer programming expertise. This course is a complete course of Embedded OS and as of now no books are written on this complete flow and life cycle. Since the most of Embedded OS concept, Implementation and environment. is generic so once you are having good hands over here you will be confident in this independent of Hardware/OS.</p> <p>Every class is backed by discussion and topic related assignments, demo by instructor and practice by you. Finally a project letting you apply most of the concepts learned throughout the course.</p>
Objective	<ul style="list-style-type: none"> ● To be aware of various trends in Embedded OS ● You will come to know about mostly used hardwares i.e Flashes, EEPROMS and developmental boards. ● After completing this course you will be able to start developing any high end (embedded system with OS) embedded project.
Platform	ARM 9 (AT91RM9200), Emertxe ARM development board ExDev9A and ExDev9A SDK. Host platform may be either Linux / Windows.
Delivery method	Instructor led, Assignments, Mini Project.
Course topics	Evolution of high end Embedded Systems, Host and Target concepts, uboot, cross compiling, porting Linux kernel and Emertxe file system on the board, debugging methods, Real time OS.

Module: 717

Module name: Linux Device Drivers

Overview	<p>This course provides a practical overview about the structure of device drivers, in-depth information about the interface between the device driver and the rest of the Linux kernel, and various practical exercises to develop and test device drivers for major devices in a Linux environment.</p> <p>It also covers the initiators and detailed insights of writing a Device Driver in Linux, focused and aligned towards the industry perspective. By the end of the module, students are expected to have made their concepts very clear and become comfortable interpreting data sheets, and writing any character device driver from software perspective</p>
Description	<p>It is not possible to connect new device types to a Linux machine without the necessary precautions. Apart from the required hardware interfaces, additional code should be added to the Linux kernel to interface between the hardware and the generic kernel routines of the I/O-subsystem. In order to write a device driver detailed knowledge is required about the internal concepts of certain parts of the Linux kernel, about the way the Linux kernel communicates with the device driver (and vice versa) and about the way a device driver handles the physical device.</p> <p>In addition, experience is needed with various types of device drivers and the way a new device driver can be added to the kernel. Upon completion of the course, candidates will understand the Linux architecture, hardware and memory management, modularization, and the layout of the kernel source, and will have practiced key concepts and skills for development of Character Drivers and good familiarity with USB and File-system drivers</p>
Objective	<ul style="list-style-type: none"> ● To provide an overview of Linux kernel internal mechanisms. ● To become proficient Linux driver developer for character based devices ● To recognize inefficient drivers, and tailor to minimize RT latency.
Platform	Linux
Delivery method	Workshop based Instructor led, Assignments, Project.
Course topics	<ul style="list-style-type: none"> ● Summarize kernel mechanisms. ● General function of the device driver: Types of device drivers, physical I/O, major and minor numbers. Loadable versus static drivers. ● Device driver types: Character drivers, block drivers and file system. ● General mechanisms: Sleep and wakeup (wait queues). Buffer allocation. Timer handling. Interrupt handling. ● Configuration and initialization: Error logging, debugging, the /proc and /dev file systems. ● Character drivers: Data transport between user mode and kernel mode, error codes, the ioctl interface. ● USB Drivers: Mini Project on live hardware ● File System and Block Drivers: Concepts and Approach to

	develop these drivers(Live Practical sessions out of scope in this duration)
--	--