System Programming Laboratory

CourseCode: MCSL27 **Credits: 0:0:1**
Prerequisites: Operating System, Linux CLI and C programming **Contact Hours: 28**
Course Coordinator: Dr. Shilpa Chaudhari

Course Contents:

This laboratory course helps students to visualize the operating system concepts through various experiments. Following are the list of experiments on the specific operating system concepts.

1. Linux kernel architecture - kernel space components, system calls
2. Kernel logging and printk
3. Key internals aspects regarding task structure
4. key internals aspects regarding CPU scheduling within the kernel
5. Key internals regarding memory management within the kernel -examine kernel segment
6. Understand and work with various dynamic kernel memory alloc/dealloc APIs - kernel page allocator, kernel slab allocator, kmalloc API
7. Kernel memory allocation - vmalloc
8. Find out how to work with key kernel synchronization primitives
9. Write high-quality modular kernel code (LKM framework) for 5.x kernels
10. Write high-quality modular kernel code (LKM framework) with parameters passing for 5.x kernels
11. Configure a kernel from source for a given requirement - Setting up the software, static analysis tools for Linux Kernel
12. Build a kernel from source- customized kernel, kernel image, customize GRUB boot loader

Reference Books:

1. Billimoria, Kaiwan N., Linux Kernel Programming: A comprehensive guide to kernel internals, writing kernel modules, and kernel synchronization. Packt Publishing Ltd, 2021.
2. Bharadwaj, Raghu. Mastering Linux Kernel Development: A kernel developer's reference manual. Packt Publishing Ltd, 2017.

CourseOutcomes(COs):

At the end of the course, the students will be able to:

1. Understand the essentials of key internals topics such as kernel architecture, memory management, CPU scheduling, and kernel synchronization (PO1,3,4)
2. Explore and analyze the operating system concepts by building the kernel from the source using most recent Long-Term Support Linux kernel (PO1,3,4)
3. Discover how to write kernel code using the Loadable Kernel Module framework (PO1,3,4)