Course Title	**Operating Systems Lab**
Course Code	**CC-311L**
Credit Hours	1
Category	Computing Core
Prerequisite	Data Structures and Algorithms
Co-Requisite	None
Follow-up	None
Course Description	Interacting with Linux Operating System: Virtualization and Hypervisors, Linux distributions. Installing Linux on Sun Virtual Box. Linux File hierarchy standard. File System Architecture: Schematic view of a standard UNIX file system. Describe the contents of boot block, superblock, inode block, and data blocks. File System Mounting: Introduction to the concept of file system mounting. Linux configuration files related to file system mounting. Linux commands like mount, umount, lsblk, blkid. Maintaining the integrity of the file system using Linux commands like fsck, e2fsck, fsck. fat, fsck. nfs. File Permissions: standard file permissions. Use of chmod and chown commands. Setting the default file permissions on a newly created file using the umask command. Special File Permissions: Concept and use of Saved SUID bit on files. Concept and use of Saved SGID bit on files and directories. Concept and use of Sticky bit on files and directories. Device files: Seven File Types in Linux and the concept of device files. Describes the contents of /dev/ directory. Terminal Attributes: Overview of Terminal Devices and a comparison between disk and terminal files. Examine current attributes of the terminal driver on a Linux machine and change them using the stty command. Hard and Soft Links: the use of hard and soft links on all UNIX-based systems. Differences between hard and soft links. Use of Linux command ln to create hard and soft links. Managing services using systemd: Introduction to Linux system daemon. Overview of systemd unit files, especially Targets Unit Files and Service Unit Files. Shell commands to manage services using systemctl. Writing/running a basic service of your own. Booting process of a Linux system: A discussion on five phases of Linux Operating system: BIOS / UEFI Initialization, Master Boot Record, Boot Loader, Kernel Initialization, init or systemd Process. Linux System Programming: Linux System Call Interface, Use of GNU gcc compiler on Linux terminal, Format of a program file on disk and its components. Viewing contents of a program file using objdump and readelf commands. Process Creation and Termination: getpid(), getppid(), fork(), exit(), wait() and execl() system calls. File management in Linux. Concept of PPFDT. Concept of input, output and error redirection. Inter-Process Communication: Linux IPC tools, Pipes, FIFOS, and Sockets. Use of pipes and fifos on a Linux terminal. Signals: Signal delivery and execution of a signal handler. Synchronous and Asynchronous signals. Standard and real-time signals. Sending signals to running processes using the kill command. Signal disposition of some important signals like SIGHUP, SIGINT, SIGKILL, SIGPIPE, SIGALARM, SIGTERM, SIGQUIT, SIGILL, SIGFPE, SIGSEGV, SIGSTOP, SIGTSTP, SIGCHLD, SIGCONT. Threads and Scheduling: Writing multi-threaded C programs using library calls from the POSIX pthread library like pthread_create(), pthread_join(), and pthread_exit(). Use of Linux schedtool command to query and change different CPU scheduling

	parameters like scheduling policy, nice value, static priority, CPU affinity, Thread synchronization using pthread_mutex_t variable and pthread_mutex_lock() and pthread_mutex_unlock() library calls, Use of Linux tools like mkfs, mke2fs, mkntfs, mkfs.fat, mkfs. minix to put a file system on a partition
Text Book(s)	Sarwar and Koretsky, Unix: The Text Book, 3rd edition, ISBN-13: 978-1-4822-3358-2.
Reference Material	