IT227 – Computer Systems Programming

Course Placement: Computer Systems Programming is a core course for second-year students of the BTech ICT program.

Course format: It is **3 hours lecture** and **2 hours lab practical** every week.

Prerequisite: IT112 and IT113 Introduction to Programming (C Programming), IT121 Digital Logic Design and Computer Organization.

Course Content: This course aims to introduce the students to system-level programming in a UNIX/Linux environment. The students will be introduced to the standard Linux commands, memory management, interacting with the operating system by making system calls for file management, file execution, process control, and inter-process communication, shell scripting, Sockets and using TCP/IP, Shell principles, exec family of functions, naming conventions, and so on. A primary goal of the course then is to train the students in a systems programming context to develop robust code.

Text Books:

- Advanced Programming in the Unix Environment, 2nd Edition, by Richard Stevens, Stephen Rago, Pearson Education Inc.
- The C Programming Language, 2nd Edition, by Brian Kernighan and Dennis Ritchie, Prentice Hall

Reference Book:

- Understanding Unix/Linux Programming: A Guide to Theory and Practice, by Bruce Molay, Prentice Hall

Assessment method:

- Lab Programming Assignments: 20%
- 1st In-Semester Exam: 15%
- 2nd In-Semester Exam: 25%
- End-Semester Exam: 30%
- Class Participation and Quizzes: 10%

Course Outcomes:

After successful completion of the course, students will have the ability to

- Understand Linux operating system commands, the concept of virtualization, steps of program compilation using GCC compiler, operating systems concepts such as process, threads and interprocess communication, and computer networking concepts.
- Acquire practical knowledge by writing complex shell scripts, developing Unix/Linux libraries, developing parallel applications using multiprocess and multithreading, performing interprocess communication in systems with shared-memory architecture using pipes and signals, and developing an application to communicate between networked computers using sockets.

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12
	X	X									X

Lecture Schedule:

SI. No.	Description	No. of Lectures
1	Introduction: System Programming introduction	2
2	OS Virtualization, Unix Shell, Basic Shell Commands, Shell Scripting	4
3	C Programming: Using the C programming language, its constructs and grammar, to create the system software	3
4	Usage of Unix C compiler GCC, compiling, linking, object files, loading, symbol resolution, shared and static libraries, debugging, and execution of system programs, makefiles	5
5	File IO: unbuffered IO (file descriptor, open, create, read and write files, file modes), buffered IO (file pointer, open, create, read, write files, file modes), advantage and disadvantage between both schemes, directories, symbolic links, permissions	3
6	Process: creation and termination of the process, process states, exec family system functions, fork to create a child process, process control, inter-process communication (IPC) using pipe	5
7	Signals: types of signals, signal actions, catching and handling signal, send/receive a signal from the process, inter-process communication (IPC) using signals	5
8	Concurrent Programming: computing and communication type: multithreading, parallel computing (shared memory), distributed computing (message passing), threads vs. process, user-level vs. kernel-level threads, POSIX threads: creation, termination, join, synchronization, critical section, mutual exclusion, mutex locks, semaphores, deadlock, reader-writer problem, dining-philosopher problem	6
9	Network Programming: Communication Layers (Network, Transport), Protocols basics: Internet Protocol (IP), TCP: connection-oriented, UDP: connectionless, standard services and assigned ports, client-server communication using sockets	6