School of Interdisciplinary Engineering and Sciences (SINES)
National University of Sciences & Technology (NUST)

Course Title: Computing for Computational Science &Engineering
Course Code: CSE- 887
Credit Hours: 3-0
Pre Requisites:

Course Introduction:

The course is intended to be self-consistent, with no prior computer skills required. The course is aimed to familiarize students with the necessary computational ideas, software, and techniques that will serve as a foundation for students to undertake other advanced courses in the MS Computational Science and Engineering degree program. Moreover, the knowledge gained will be beneficial for conducting research in the various fields of computational science and engineering. The main idea of the course is to give the student hands-on experience in writing a simple software package that eventually can be implemented on a parallel computer architecture. The course is divided into three modules. In the first module, students will be introduced to the Linux working environment which is required to work on supercomputing machines and cloud servers. In the second module, Python programming will be introduced, which has become a leading language in the scientific computing community. In the third module, the students will be introduced to the main parallel programming techniques and the common software packages/libraries.

Learning Objectives:

The learning objectives of the course are:

1. **Understanding the basic concepts of Linux**: Students will learn about the history of Linux, its architecture, and its role in the modern computing environment.
2. **Familiarity with Linux command-line interface**: Students will become familiar with the Linux command line, including navigating the file system, manipulating files and directories, and executing commands.
3. **Knowledge of Linux system administration**: Students will learn about the Linux file system, user and group management, and process management, as well as network configuration and basic security practices.
4. **Understanding of Linux package management**: Students will learn how to manage software packages in Linux, including installing, updating, and removing software.
5. **Practical experience with Linux**: Students will gain practical experience using Linux by completing hands-on exercises and assignments.
6. **Knowledge of Linux scripting**: Students will learn basic scripting in Linux using Bash, and be able to automate repetitive tasks and create simple shell scripts.
7. **Understanding of basic programming concepts**: Students will learn about programming concepts such as variables, data types, conditionals, loops, functions, and basic data structures.
8. **Proficiency in the Python programming language**: Students will gain practical experience in coding using Python, and will learn the syntax, libraries, and tools associated with the language.
9. **Ability to solve simple programming problems**: Students will be able to write simple programs to solve basic problems, and will gain an understanding of how to decompose a problem into smaller, more manageable sub-problems.
10. **Familiarity with software development tools**: Students will become familiar with the tools commonly used in software development, such as text editors, version control systems, and integrated development environments.
11. **Good programming practices**: Students will learn about good programming practices, such as writing clean code, testing and debugging, and documenting code.

12. **Critical thinking and problem-solving skills**: Students will develop critical thinking and problem-solving skills, as programming involves breaking down a problem into smaller parts and using logic to solve it.
13. **Understanding of parallel computing concepts**: Students will learn about parallel computing, including the different types of parallelism and architectures, and the benefits and challenges of parallel programming.
14. **Practical experience in parallel programming**: Students will develop skills in writing parallel programs using Python and parallel computing libraries such as MPI and OpenMP.

Weekly Lecture Plan

Week 1: Introduction to Linux and the Command Line

- Overview of Linux and its history
- Advantages of using Linux over other operating systems
- PuTTY and WinSCP
- Basics of the command line interface (CLI)
- Basic directory operations in Linux (pwd, ls, cd, mkdir, rmdir)
- Hands-on practice lab/Assignment

Hands-on practice lab to reinforce concepts

Week 2: Linux basic file operations

- Basic file operations in Linux (touching files, vi editor)
- Displaying files (cat, more, less, head, tails, which)
- Finding files (find command)
- Other important common file operations (cp, mv, rm, cut, paste, word count)

Hands-on practice lab to reinforce concepts

Week 3: Linux File System and Permissions

- Understanding the Linux file system
- File ownership and permissions in Linux (chmod)
- User and group management in Linux
- Special file types and permissions
- grep command

Hands-on practice lab to reinforce concepts

Week 4: Advanced Linux Concepts and Scripting

- I/O Redirections
- Pipes
- Shell scripting in Linux
- Understanding environment variables in Linux

Hands-on practice lab to reinforce concepts

Week 5: Software management in Linux

- Installing software from Compressed Archives
- tar, gzip, gunzip commands
- Configuration Script and Make File
- Compiling software
- Red Hat Package Manager (RPM) and Debian Package Manager (DEB)

Hands-on practice lab to reinforce concepts

Week 6: Introduction to Python

- Introduction to Python and its applications
- Installing Python and setting up the development environment
- Basic Python syntax, print, variables, data types
- Working with the Python shell and running Python programs

Hands-on practice lab to reinforce concepts

Week 7: Conditionals, Controls and Loops

- Conditional expressions
- Control flow statements: if, else, and elif
- Loops: for and while

Hands-on practice lab to reinforce concepts

Week 8: Lists, Sets, Tuples, Dictionaries

- Lists: creating and modifying lists, accessing elements, and slicing
- Tuples: creating and using tuples
- Dictionaries: creating and accessing dictionaries, and iterating through keys and values

Practice exercises to reinforce concepts

Week 9: Mid-Semester Exam

Week 10: Strings and Files

- String manipulation: creating and formatting strings, and working with string methods
- File handling: opening, reading, and writing files

Practice exercises to reinforce concepts

Week 11: Functions and Exception Handling

- Functions: defining and calling functions, parameters, and return values
- Exception handling: using try and except statements

Practice exercises to reinforce concepts

Week 12: Introduction to Object-Oriented Programming in Python

- Introduction to OOP concepts: classes, objects, inheritance, polymorphism, encapsulation
- Creating classes and objects in Python
- Instance variables and methods
- Class variables and methods
- Inheritance and polymorphism in Python

Practice exercises to reinforce concepts

Week 13: Introduction to Parallel Computing

- Introduction to parallel computing concepts: concurrency, parallelism, speedup, efficiency, scalability, Amdahl's Law
- Parallel hardware architectures: shared memory, distributed memory, hybrid

Week 14: Parallel programming models

- Parallel programming models: shared memory, message passing, hybrid
- Introduction to parallel programming in Python: multiprocessing, multithreading

Week 15: Parallel Programming with OpenMP

- Introduction to OpenMP for shared memory parallel programming
- OpenMP programming model: threads, work sharing, synchronization, affinity
- Design and implementation of parallel algorithms using OpenMP

Practice exercises to reinforce concepts

Week 16: Parallel Programming with CUDA

- Introduction to CUDA for GPU programming
- CUDA programming model: kernels, threads, blocks, memory hierarchy
- Design and implementation of parallel algorithms using CUDA

Practice exercises to reinforce concepts

Week 17: Course Recap

Week 18: End Semester Exam

Course Learning Material:

- Lecture slides
- "*Linux in easy steps*" by Mike McGrath | 2021
- "*The Linux Programming Interface: A Linux and UNIX System Programming*" by Michael Kerrisk | 2010
- "*Python Crash Course*", by Eric Matthes | 2016
- "*Learn to Code by Solving Problems*", by Daniel Zingaro | 2021
- "*Hands-On GPU Programming with Python and CUDA*", by Brian Tuomanen | 2018

Grading

Nature of Examination	**Weightage**
Assignments	5-10%
Quizzes	10-15%
Mid Semester Exam	30-40 %
End Semester Exam	40 - 50%