**Concepts of Operating Systems&Software Development Methodologies**

PG-DAC August 2025

**Duration: 74 hours (50 theory hours + 24 lab hours)**

**Evaluation:** 100 marks

**Weightage:** Theory exam – 40%, Lab exam – 30%, Internals – 30%

**Concepts of Operating Systems**

**Duration: 26 hours (18 theory hours + 8 lab hours)**

**Objective:** To introduce Operating System concepts with Linux environment, and to learn Shell Programming.

**Prerequisites:** Knowledge of computer fundamentals

**Evaluation: 35 marks (CCEE:15 +Lab exam:10 + Internals:10)**

**Text Books:**

* Operating Systems Principles by Abraham Silberschatz, Peter Galvin& Greg Gagne / Wiley
* Unix Concepts and Applications by Sumitabha Das / McGraw Hill

**References:**

* Modern operating Systems by Andrew Tanenbaum & Herbert Bos/ Pearson
* Principles of Operating Systems by Naresh Chauhan / Oxford University Press
* Beginning Linux Programming by Neil Matthew & Richard Stones / Wrox
* Operating System : A Design-Oriented Approach by Charles Crowley / McGraw Hill

# (Note: Each Session is of 2 hours)

**Session 1: Introduction to OS**

**Lecture:**

* What is OS; How is it different from other application software; Why is it hardware dependent?
* Different components of OS
* Basic computer organization required for OS.
* Examples of well-known OS including mobile OS, embedded system OS, Real Time OS, desktop OS server machine OS etc. ; How are these different from each other and why
* Functions of OS
* User and Kernel space and mode; Interrupts and system calls

***No Lab***

**Session 2: Introduction to Linux**

**Lecture:**

* Working basics of file system
* Commands associated with files/directories & other basic commands. Operators like redirection, pipe
* What are file permissions and how to set them?
* Permissions (chmod, chown, etc); access control list; network commands (telenet, ftp, ssh, sftp, finger)
* System variables like – PS1, PS2 etc. How to set them

***Shell Programming***

* What is shell; What are different shells in Linux?
* Shell variables; Wildcard symbols
* Shell meta characters; Command line arguments; Read, Echo

**Lab: (4 hours)**

* Working with various OS commands
* Shell programs related to Session 2

**Session 3: Shell Programming**

**Lecture:**

* Decision loops (if else, test, nested if else, case controls, while…until, for)
* Regular expressions; Arithmetic expressions
* More examples in Shell Programming

**Lab: (4 hours)**

* Shell Programs related to Session 3

**Sessions 4 & 5: Processes**

**Lecture:**

* What is process; preemptive and non-preemptive processes
* Difference between process and thread
* Process management; Process life cycle
* What are schedulers – Short term, Mediumterm and Long term.
* Process scheduling algorithms – FCFS, Shortest Job First, Priority, RR, Queue. Belady’s Anomaly
* Examples associated with scheduling algorithms to find turnaround time to find the better performing scheduler.
* Process creation using fork; waitpid and exec system calls; Examples on process creation; Parent and child processes
* Orphan and zombie processes

***No Lab***

**Sessions 6 & 7:**

**Lecture:**

***Memory Management***

* What are different types of memories; What is the need of Memory management
* Continuous and Dynamic allocation
* First Fit, Best Fit, worst Fit
* Compaction
* Internal and external fragmentation
* Segmentation – What is segmentation; Hardware requirement for segmentation; segmentation table and its interpretation
* Paging – What is paging; hardware required for paging; paging table; Translation look aside buffer
* Concept of dirty bit
* Shared pages and reentrant code
* Throttling
* IO management

***No Lab***

**Session 8:**

**Lecture:**

***Virtual Memory***

* What is virtual memory
* Demand paging
* Page faults
* Page replacement algorithms

***No Lab***

**Session 9:**

**Lecture:**

***Deadlock***

* Necessary conditions of deadlock
* Deadlock prevention and avoidance
* Semaphore
* Mutex
* Producer consumer problem
* Dead-lock vs Starvation

***No Lab***

**Software Development Methodologies**

**Duration: 48hours** (32theory hours + 16lab hours)

**Objective:** To build knowledge of Software development methodologies.

**Evaluation: 65 marks (CCEE: 25 + Lab exam: 20 + Internals: 20)**

**Reference Books:**

* Software Engineering by Chandramouli / Pearson
* Software engineering by Ian Sommerville / Pearson
* Object-Oriented Analysis and Design Using UML - An Introduction to Unified Process and Design Patterns by Mahesh P. Matha / PHI
* Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin / Prentice Hall
* The Mythical Man-Month: Essays on Software Engineering by Frederick P. Brooks Jr. / Addison Wesley
* User Stories Applied: For Agile Software Development by Mike Cohn / Addison Wesley
* DevOps: Continuous Delivery, Integration, and Deployment with DevOps by SricharanVadapalli/ Packt
* Git for Teams by Emma Westby / O’Reilly

# (Note: Each Session is of 2 hours)

**Git (4 hours)**

**Session1**

**Lecture**

* Developing an application in a team
* Issues developers face when working in a team
* Introduction to code versioning system
* History of code versioning system
  + Different tools available for versioning
  + Software development workflow
* Introduction to git
* Introduction to git repository and git structure
* Adding code to git
* Creating and merging different git branches

**Lab**

* Create a local git repository
* Commit the initial code
* Update the code
* Use git commands to
  + Get the updated files
  + List the changes
  + Create branch
  + Merge branch

**Software Engineering& DevOps (28 hours)**

**Sessions 2, 3, 4& 5**

**Lecture**

* Introduction to software engineering
  + Software Process
  + Software Process Model
  + Software Product
* Importance of Software engineering
* Software Development Life Cycles
* Requirements Engineering
  + Types of Requirements
  + Steps involved in Requirements Engineering
  + Requirement Analysis Modelling
* Design and Architectural Engineering
  + Characteristics of Good Design
  + Function Oriented vs Object Oriented System
  + Modularity, Cohesion, Coupling, Layering
  + Design Models
  + UML
* Coding
  + Programming Principles
  + Coding Conventions
* Object Oriented Analysis and Design

***No Lab***

**Sessions6, 7 &8**

**Lecture**

* Introduction to Agile development model
* Agile development components
* Benefits of Agile
* Introduction to different tools used for agile web development
* Scrum and Extreme Programming
* Introduction to AtlassianJira
  + Add Project
  + Add Tasks and sub-tasks
  + Create sprints with tasks
* Case study of developing web application using agile methodology

***No Lab***

**Sessions9&10**

**Lecture**

* Introduction to DevOps
* DevOps ecosystem
* DevOps phases
* Introduction to containerisation
* Introduction to docker
* Creating docker images using Dockerfile
* Container life cycle

**Lab**

* Install and configure docker
* Create docker image using Dockerfile
* Start docker container
* Connect to docker container
* Copy the website code to the container
* Use docker management commands to
  + List the images
  + List the containers
  + Start and stop container
  + Remove container and image

**Session 11**

**Lecture**

* Introduction to YAML
* Introduction to Docker Swarm and Docker Stack
* Introduction to Kubernetes
* Creating Kubernetes cluster
* Creating service in Kubernetes
* Deploying an application using dashboard

**Lab (4 hours)**

* Configure Kubernetes
* Configure Kubernetes Dashboard
* Setup a Kubernetes cluster
* Access application using Kubernetes service
* Deploy the website using Dashboard

**Testing& Integration (16 hours)**

**Session 12**

**Lecture**

* Introduction to software testing
* Why testing code is important
* Verification and validation
* Quality Assurance vs Quality Control vs Testing
* Principles of software testing

**Assignment**

* Read more testing concepts used in the industry

**Session 13**

**Lecture**

* Introduction to STLC and V Model
* Types of testing: manual and automation
* Tools used for automation testing
* Introduction to testing methods: white-box, black-box and grey-box
* Introduction to functional testing: (\* students are supposed to learn the concepts)
* Introduction to non-functional testing: (\* students are supposed to learn theconcepts)

**Assignment**

* Create a test plan for project
* Document the use cases
* Create test case document for different sprints (designed in SE)

**Sessions14& 15**

**Lecture**

* Introduction to Selenium (use Eclipse IDE)
* Load web driver
* Create selense commands: locators: by ID, name, class, tag name, XPath
* Add interactions: text box, radio button selection, check box selection, drop down item selection, keyboard actions, mouse actions, multi select

**Lab**

* Download and configure Selenium
* Create a test suite
* Add commands and interactions

**Session 16**

**Lecture**

* Introduction to delivery pipeline
* Introduction to Jenkins
* Jenkins management
* Adding slave node to Jenkins
* Building a delivery pipeline
* Selenium integration with Jenkins

**Lab**

* Install and configure Jenkins
* Build a pipeline job using Jenkins
* Create a maven project for Selenium
* Add Selenium test suite in the project
* Integrate it with Jenkins