MODULE DESCRIPTOR FORM

Module Information				
Module Title	**LINUX OPERATING SYSTEM**		**Module Delivery**	
Module Type	**CORE**		☒ **Lecture** ☒ **Practical**	
Module Code	**IT3201**			
ECTS Credits	**6**			
SWL (hr/sem)	**150**			
Module Level	UG III		**Semester of Delivery**	2
Administering Department	Information Technology		**College**	College of Sciences
Module Leader	**Ali Abdulhussein Ibrahim**		**e-mail**	**ali.abdulhussein@uowa.edu.iq**
Module Leader's Acad. Title	**Asst. Lecturer**		**Module Leader's Qualification**	MS.c
Module Tutor	**Ali Abdulhussein Ibrahim**		**e-mail**	ali.abdulhussein@uowa.edu.iq
Peer Reviewer name	Asst. Lect Karrar Sadiq Mohsen		**e-mail**	karar.sadeq@uowa.edu.iq
Review Committee Approval	2026-2025		**Version Number**	V1

Relation With Other Modules			
Prerequisite module	-	**Semester**	-
Co-requisites module	-	**Semester**	-

جامعة وارث الأنبياء
كلية العلوم
قسم تكنولوجيا المعلومات

[illegible]
أ.م.د. حيدر محمد علي الغانمي
ر.ق
2026/2025

جامعة وارث الأنبياء
عمادة كلية العلوم

[illegible]
أ.م.د. شيماء حسين نوفل
٢٠٢٥ - ٢٠٢٦م

Department Head Approval **Dean of the College Approval**

Module Aims, Learning Outcomes and Indicative Contents	
Module Aims	1. Understand the rationale behind the current design and implementation Linux Operating system 2. Provides an introduction to Kernel Driver for the Linux operating system. 3. learn about Linux shell scripting and System Programming concepts. 4. learn the basics of File IO for Linux. 5. Understand the basics of Process Management, process forking, and Linux daemons. 6. learn the basics of Threading and Multi Thread Synchronization in Linux system programming. 7. Identify the deadlock problem in multi-threading program. 8. Expose the role of CPU scheduling algorithms in OS. 9. Understand different approaches for memory management.
Module Learning Outcomes	1. Configure, build and deploy the Linux kernel and root file system from source. 2. Use System Programming concepts to develop components of a Linux Embedded System, including kernel and root file system. 3. Build own custom Embedded bash programs through programming assignments. 4. Describe Threading and Multi Thread Synchronization in Linux system 5. Explain inter-thread and inter-process communication using semaphore 6. Explain what deadlock is in relation to operating systems; 7. Discuss deadlock prevention, avoidance, and the differences between each; 8. Describe deadlock detection and recovery. 9. Discuss CPU scheduling and its relevance to operating systems; 10. Explain the general goals of CPU scheduling; 11. describe the differences between preemptive and non-preemptive 10. scheduling; and discuss four CPU scheduling algorithms. 12. Explain the memory hierarchy;

	13. Discuss how the operating system interacts with memory; 14. Describe how virtual memory works; 15. Discuss three algorithms for dynamic memory allocation; 16. Explain methods of memory access; and describe paging and page replacement algorithms.
Indicative Contents	Indicative content includes the following. 1. Introduction to Linux System Programming It deals with a binger to high level Linux shell scripting and System Programming concepts. It also identifies the basics of an Embedded Linux tool chain. 2. Deadlock and Synchronization problems Deadlock is a paralyzing process state resulting from improper process management, and synchronization management. In deadlock, processes are blocked as they compete for system resources or only communicate with each other. Although it cannot be guaranteed that deadlock may be avoided 100% of the time, it is important to know how to avoid the deadlocked state and how to recover from it once it has been achieved. It needs the previous two units of Processes and Threads and synchronization when discussing Deadlock. Firstly, deadlock is discussed by establishing a working definition and the conditions in which it presents itself. Then, clarify how to prevent and avoid deadlock. Finally, we will learn about deadlock detection, as well as methods for recovering from a deadlocked state. Processes and Threads. 3. CPU Scheduling Central Process Unit (CPU) scheduling deals with having more processes/threads than processors to handles those tasks, meaning how the CPU determines which jobs it is going to handle in what order. A good understanding of how a CPU scheduling algorithm works is essential to understanding how an Operating System works; a good algorithm will optimally allocate resources, allowing an efficient execution of all running programs. A poor algorithm, however, could result in any number of issues, from process being "starved out" to inefficient executing, resulting in poor computer performance. A number of common types of CPU scheduling, such as preemptive and non-preemptive will be

	explained in UNIX-based Operating Systems. 4. Memory Management Memory is the key aspect that keeps the computer running smoothly. It presents in various forms throughout the entire computer system. It is essential to have a solid understanding of the role memory plays so that you are able to efficiently use memory in your programs, as well as understand what is going on "under the hood" should a problem arise. The role of memory in an Operating System will be discussed, first with an overview of the memory hierarchy and how memory and the OS interact with each other. Next, we will move on to discussing how memory is allocated for different purposes. Finally, we will discuss the two main topics regarding memory access: segmentation and paging.

Learning and Teaching Strategies	
Strategies	The learning and teaching strategies for studying the Operating systems in an IT department comprises a balanced strategy of theoretical understanding and practical application. Lectures, interactive discussions, practical exercises that come from theoretical foundation and seminars. Group work, ring discussion and projects enable hands-on experience with operating systems. Online resources, assessments, and feedback aid in reinforcing learning. Virtual labs and consistent learning that support the practical skills development and staying updated with cutoff trends. These strategies ensure a comprehensive understanding of operating systems and their impact in the IT field.

Student Workload (SWL)			
Structured SWL (h/sem)	60	**Structured SWL (h/w)**	4
Unstructured SWL (h/sem)	87	**Unstructured SWL (h/w)**	6
Total SWL (h/sem)	147 + 3 final = 150		

Module Evaluation					
		Time/Number	Weight (Marks)	Week Due	Relevant Learning Outcome
Formative assessment	Quizzes	5	10% (10)	5,10	
	Assignments	1	5% (5)	4,12	
	Lab/Project	5	15% (15)	مستمر خلال الفصل	
	Report	1	10% (10)	13	
Summative assessment	Midterm Exam	2hr	10% (10)	7	
	Final Exam	3hr	50% (50)	16	
Total assessment			100% (100 Marks)		

Delivery Plan (Weekly Syllabus)	
	Material Covered
Week 1	Introduction to Linux Operating System: concepts and functionality
Week 2	Linux Operating Systems Overview (Shell, Linux (Ubuntu) Windows, Mobile, Real Time)
Week 3	Synchronization problems (Read-write Problem, Dining philosopher problem, starvation Problem)
Week 4	Deadlock, Definition, Conditions, Resources allocation graph
Week 5	Deadlock Prevention and Avoidance
Week 6	Banker Algorithm
Week 7	Scheduling General Objectives
Week 8	Pre-emptive v. non-preemptive, Scheduling policy goals
Week 9	Scheduling policies: First in First out, Shortest Job Next, Preemptive shortest job next 5
Week 10	Scheduling policies: Priority Scheduling, Round Robin
Week 11	Types of scheduler shortest term scheduler, long term scheduler, multi- level queue scheduler, Multilevel feedback queue
Week 12	Overview of Memory Management, Memory Hierarchy
Week 13	OS Interaction with Memory Levels, Virtual Memory
Week 14	Memory Access, Allocating Memory
Week 15	Paging and Segmentation
Week 16	Preparatory week before the final Exam

Delivery Plan (Weekly Lab. Syllabus)	
	Material Covered
Week 1	Lab 1: Bash and Bash scripts
Week 2	Lab 2: Shell built−in commands, Common shell programs
Week 3	Lab 3: Bash startup files, build Interactive shells
Week 4	Lab 4: Conditionals, Shell arithmetic, loops
Week 5	Lab 5: Use Mutex to solve Read Write synchronization problem in Linux Operation system
Week 6	Lab 6: Deadlock Example using threads and mutex
Week 7	Deadlock Detection and Recovery (banker algorithm), identify the input and output arguments
Week 8	Lab 8: Implementation of banker algorithm for deadlock detection build and Implement project for each student
Week 9	Lab 9: Design and implements First Come First Services Scheduling algorithm
Week 10	Lab 10: Design and implements Shortest Job Next Scheduling algorithm
Week 11	Lab 11: Design and implements Preemptive Shortest Job Next Scheduling algorithm
Week 12	Lab 12: Design and implements Priority Scheduling algorithm
Week 13	Lab 13: Design and implements Round Robin Scheduling algorithm
Week 14	Lab 14: Design and implementation of Memory allocation algorithm First Fit
Week 15	Lab 15: Design and implementation of Memory allocation algorithm Best Fit

Learning and Teaching Resources		
	Text	**Available in the Library?**
Required Texts	Operating System Concepts, ABRAHAM SILBERSCHATZ, PETER BAER GALVIN, GREG GAGNE, 9 EDITIONS, Copyright! © 2013, 2012, 2008 John Wiley& Sons	Yes
Recommended Texts	Pro Bash Programming Scripting the GNU/Linux Shell, Chris F.A. Johnson,2009 Bash Guide for Beginners, Machtelt Garrels, 2006. Linux for Beginners-Josh Thomson ,2017	No
Websites		

APPENDIX:

GRADING SCHEME				
Group	**Grade**	**Mark**	**Marks (%)**	**Definition**
Success Group (50 - 100)	**A -** Excellent	Excellent	90 - 100	Outstanding Performance
	B - Very Good	Very Good	80 - 89	Above average with some errors
	C - Good	Good	70 - 79	Sound work with notable errors
	D - Satisfactory	Fair / Average	60 - 69	Fair but with major shortcomings
	E - Sufficient	Pass / Acceptable	50 - 59	Work meets minimum criteria
Fail Group (0 – 49)	**FX** – Fail	Fail (Pending)	(45-49)	More work required but credit awarded
	F – Fail	Fail	(0-44)	Considerable amount of work required
Note:				

Note: Marks Decimal places above or below 0.5 will be rounded to the higher or lower full mark (for example a mark of 54.5 will be rounded to 55, whereas a mark of 54.4 will be rounded to 54. The University has a policy NOT to condone "near-pass fails" so the only adjustment to marks awarded by the original marker(s) will be the automatic rounding outlined above.