

COURSE DESCRIPTION

Dept., Number	CS351	Course Title	Systems Programming
Semester hours	3	Course Coordinator	Matthew Bauer, Senior Lecturer

Current Catalog Description

Examines the components of sophisticated multi-layer software systems-including device drivers, systems software, applications interfaces, and user interfaces. Explores the design and development of interrupt-driven and event-driven software. Prerequisites: CS 331, CS 350. (2-2-3)

Textbook

Bryant, Randal E., and David O'Hallaron. Computer Systems: A Programmer's Perspective. PrenticeHall, 2003

References

Kernighan, Brian W., and Dennis M. Ritchie. The C Programming Language, 2nd Edition. Prentice Hall, 1988.

Rochkind, Marc J. Advanced UNIX Programming. Addison-Wesley, 2004

<http://www.cs.iit.edu/~lee/cs351/resources.shtml>

Course Outcomes

Students should be able to:

- Define the concept and role of a process in a modern operating system
- Describe the key abstractions an operating system provides to running processes
- Describe the function, usage, and operation of system calls related to process management, memory management and I/O
- Explain exceptional control flow, including:
 - Hardware interrupts
 - Software exceptions / Traps
 - Signals and signal handling
- Describe the essential operation of a modern MMU from a programmer's standpoint, including:
 - Caching and the TLB
 - Segmentation and paging for virtual memory
- Explain the operation of various memory allocation methods, including:

- Implicit allocation (garbage collection)
- Explicit allocation (malloc/free, reference counting, etc.)
- Describe, utilize, and implement a dynamic memory allocation API.
- Describe and utilize the system-level I/O API of a modern operating system, including:
 - File descriptors
 - File I/O
 - Buffered I/O
 - Interprocess communication
- Describe and utilize a low-level socket based networking API. This should include:
 - Client / Server model
 - Internetworking
 - Berkeley sockets
- Describe, design and utilize concurrent programming APIs, including:
 - POSIX Threads
 - Re-Entrant code
 - Synchronization primitives

Relationship between Course Outcomes and Program Outcomes

The following Program Outcomes are supported by the above Course Outcomes:

- a. An ability to apply knowledge of computing and mathematics appropriate to the discipline
- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
- c. An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs
- i. An ability to use current techniques, skills, and tools necessary for computing practices.
- j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices

Prerequisites by Topic

CS 331 Data Structures, CS350 – C/Assembly Programming

Major Topics Covered in the Course

1. Introduction and Syllabus, Course Overview	2 hours
2. Assembly review / x86 Assembly Primer	2 hours
3. C: Language basics, Pointers, Arrays, and Structures	8 hours
4. Processes and the OS, Process management	6 hours
5. Exceptional Control Flow (signals, signal handling, etc.)	4 hours
6. Practical: Programming a UNIX shell	2 hours
7. Caching and Virtual Memory	4 hours
8. Dynamic Memory Management	6 hours
9. Practical: Implementing malloc	2 hours
10. UNIX System Level I/O	4 hours
11. Interprocess Communication (pipes, message queues, shared memory, etc.)	5 hours
12. Berkeley sockets API	5 hours
13. Practical: A Concurrent Server	2 hours
14. POSIX Threads API	4 hours
15. Midterm, Recap & Review	4 hours
Final Exam	-
	60 hours

Assessment Plan for the Course

End of every semester Course Objective Assessments by CS department. End of semester Course Evaluations by IIT. Reviewed every Spring semester by CS Undergraduate Studies Committee for possible updates in the following Fall. Once every 4-5 years a detailed review of all materials for the course is made by the CS Undergraduate Studies Committee.

How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

See the assessment discussion under Criterion 4

For a computer science program

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms	1		Software design	1	
Data structures			Concepts of programming	1	

		languages		
--	--	-----------	--	--