EECE 432– Operating Systems (3 credits)

Catalog description:

This course covers the principles of operating systems and systems programming. The topics discussed in class are processes, threads, concurrency and synchronization, scheduling, deadlocks, memory management, file systems, i/o devices, parallel and distributed systems, and security. The course will be accompanied with hands on assignments involving contemporary linux kernels.

Areas: Software engineering

Required or Elective:

Elective for CCE / ECE
Level: Second year, third year, senior or graduate standing

Prerequisites:

By topic: EECE 321 and EECE 330.

Textbooks:

- Modern operating systems. Andrew Tanenbaum. 2009, Pearson-Prentice Hall.
- Operating system concepts. Silberschatz, Galvin, and Gagne. 2008. John-Wiley.

Course objectives:

The objectives of this course are to give students:	***Correlates to Program Educational Objectives***
Knowledge and practice of operating system concepts.	**1,2,4**
Knowledge of a contemporary operating system kernel and practice on modifying kernel code.	**1,3,4**
Knowledge of concurrency and system programming.	**1,2,3**
Experience in building and enhancing large scale system software.	**2,3,4**

Topics

No.	**Subjects covered**	**50 Min. Lectures**
1	Overview of operating systems	2
2	Processes	3
3	Threads	3
4	Scheduling	2
5	Concurrency	3

6	Deadlocks	2
7	Memory management	2
8	Virtual memory	3
9	File systems	2
10	Distributed file systems	2
11	Input/Output devices	2
12	Security	2
13	Parallel, distributed and multiprocessor systems	2

Class/laboratory schedule

a) Three 50-minute lectures per week or two 75 minutes per weel.
b) Use of computer lab or personal computer is needed for working on the projects.

Course outcomes:

At the end of the course students should be able to:	***Correlates to Program Outcomes***		
	H	**M**	**L**
1. Understand operating system concepts.	**a, e, g, k**	**M**	**N**
2. Read and understand kernel code.	**a, c, e, k**	**m, n**	**J**
3. Build a given OS kernel from source code	**a, c, e, k**	**m, n**	**J**
4. Differentiate between user applications, kernel functionalities, and hardware system services	**a, e, g, k**	**M**	**N**
5. Modify kernel code to add/change functionality	**a, c, e, k**	**m, n**	**J**
6. Understand solutions for classical concurrency problems	**a, b, k**	**m, n**	**J**
7. Understand deadlocks and race conditions	**a, b, k**	**m, n**	**J**
8. Find and resolve concurrency issues (deadlock, race conditions,…) in computing systems.	**a, b, k**	**m, n**	**J**
9. Work in teams	**a, b, d**	**G**	**E**
10. Use productivity tools	**a, b, d**	**G**	**E**
11. Better operate, configure, and use computing machines.	**a, b, j, k**		
12. Recover a system from a software failure state	**j, k**		**A**

Resources for the course:

Books, arctiles, publications, online material

Evaluation:

1. Class participation and homework: 10 %
2. Exams: 50 %
3. Projects: 40 %

Students will work in teams to finish three projects. The first and second project will consist on modifying the kernel of an operating system to customize a specific behavior. The third project is to examine a case study or build a module from scratch where students get exposed and focus on one specific operating system concept. Students may work on ideas of their own after consulting with the course instructor.

Professional component:

Engineering topics: 80%
General education: 10%
Mathematics and basic sciences: 10%

Computer usage:

Students will work on linux and use C/C++ as their programming language.

Person(s) who prepared this description and date of preparation:

Fadi Zaraket, Oct 2009