**Department of Computer and Information Science**
**Descriptions of Elective and Graduate Courses for Fall 2010**
**Updated 5 April 2010**

## Undergraduate Elective Offerings

* denotes special topics offering
CSci 323-01: Systems of Programming, Dr. Tobin Maginnis
CSci 524-01: Distributed Operating System Design, Dr. Tobin Maginnis
CSci 555-01: Functional Programming, Dr. Conrad Cunningham
CSci 561-01: Computer Networks, Dr. Paul Ruth
CSci 581-01: *Scripting Languages, Dr. Dawn Wilkins

## Graduate-level Offerings

* denotes special topics offering
Engr 691-62: *Advanced Graphics, Dr. Phillip Rhodes
Engr 692-66: *Machine Learning, Dr. Yixin Chen

**CSci 323-01: Systems of Programming**

Instructor: Dr. Tobin Maginnis (ptm@pix.cs.olemiss.edu)
Associate Professor, Computer and Information Science

Time/Room: 10:00 MWF, Weir 106

Prerequisites:

Textbook:

Description:

Review of open source software with an in-depth analysis of the Unix operating system design with an emphasis on component interactions.
Operating system administration issues including installation, configuration, and performance tuning.
Also, open source software development process will be used.

**CSci 524-01: Distributed Operating System Design**

Instructor: Dr. Tobin Maginnis (ptm@pix.cs.olemiss.edu)
Associate Professor, Computer and Information Science

Time/Room: 2:00 MWF, Weir 235

Prerequisites:

Textbook:

Description:

**CSci 555-01: Functional Programming**

Instructor: Dr. Conrad Cunningham **(**cunningham@cs.olemiss.edu**)**
Professor and Chair, Computer and Information Science

Time/Room: 8:00 TuTh, Weir 235

Prerequisites: Either CSci 211 and Math 301 or graduate standing

Catalog Description:

The principles and techniques of programming with functions. Purely functional programming languages; recursion; higher-order functions; reduction models; strictness; type systems; list operations; infinite data structures; program synthesis and transformation.

Textbook: H. Conrad Cunningham. Notes on Functional Programming with Haskell, 2007. See http://www.cs.olemiss.edu/~hcc/csci555/notes/.\

(tentative) Bryan O'Sullivan, John Goerzen, and Don Stewart. Real World Haskell, O'Reilly Media, 2009. A free online version, licensed under a Creative Commons Attribution-Noncommercial 3.0 License, is available at http://book.realworldhaskell.org/.

Course Outcomes:

Upon successful completion of the course, the students:

1. know and understand the fundamental concepts, techniques, terminology of the functional programming paradigm

2. know and understand the syntax, semantics, and pragmatics of the polymorphically statically typed, lazily evaluated, purely functional programming language Haskell

3. can analyze problems and apply functional programming concepts and techniques to develop appropriate Haskell programs to solve the problems

4. can reason about and manipulate Haskell programs mathematically.

Instructor's Comment:

Of the more than 25 different course topics I have taught, Functional Programming is probably my favorite. I developed CSci 555 during my second year as a faculty member and will be teaching it for the 11th time in Fall 2010.

I started using the Haskell (http://www.haskell.org) programming language during the third offering of the course. During that semester I drafted an extensive set of lecture notes that evolved with subsequent offerings into one of the "textbooks" for the course.

Haskell is a language platform that has supported much of the type system and functional programming language research during the past 20 years. It is an environment that has been used to

investigate issues as wide-ranging as generic programming, concurrency (e.g., software transactional memory), data parallelism, reactive programming, metaprogramming, domain-specific languages, and programming theory.

By mastering functional programming using Haskell, I have improved my programming skills overall, regardless of whether the language I use is Java, Ruby, Scala, or Haskell. Language features such as higher-order functions and list comprehensions, which I first experienced in Haskell, now appear in languages such as Python, Ruby, C#, and Scala.  Also practical techniques like Google's Map-Reduce approach arose from the functional programming community.

I invite computer science undergraduate and graduate students to join me in this study in Fall 2010.

**CSci 561-01: Computer Networks**

Instructor: Dr. Paul Ruth
Assistant Professor, Computer and Information Science

Time/Room: 1:00 MWF, Weir 235

Prerequisites:

Textbook:

Description:

**CSci 581-01: Special Topics in Programming, (Scripting Languages)**

Instructor: Dr. Dawn Wilkins
Associate Professor, Computer and Information Science

Time/Room: 1:00 TTh, Weir 235

Prerequisites: Either CSci 211 and 223 and Senior Standing; or Graduate Standing

Textbook: None

Description:

In recent years there has been increased interest in scripting languages. Traditional programming languages such as C, Java, and others were designed to allow the development of complex programs which involve data structures and algorithms from the ground up. Scripting languages are typically use to connect (glue) together components that have previously been developed. Normally scripting languages are typeless and interpreted, whereas traditional programming languages are typed and compiled. There are advantages and disadvantages of each approach, and they complement each other. In order to perform tasks efficiently, computer scientists need to understand both paradigms and be comfortable programming in the contrasting environments.

Defining exactly what is and what is not a scripting language is a bit tricky. But in this class we will assume a loose interpretation and consider a wide range of scripting languages and their uses. We will explore Bash shell scripting and unix utilities such as grep, awk, sed, and regular expressions first. Then we will consider Perl (a general-purpose scripting language). We will also survey a wide range of scripting languages, including python, Ruby, Python, Tcl, JavaScript, PHP and others.

No textbooks will be required--material for each topic will be available online, but a few books will be recommended for those who prefer to have a hardcopy to study.

**Engr 691-62: Special Topics in Engineering Science (Advanced Graphics)**

Instructor: Dr. Phillip Rhodes
Assistant Professor, Computer and Information Science

Time/Room: 4:00 TTh, Weir 106

Prerequisites:

Textbook:

Description:

This course will examine a variety of advanced topics in Computer Graphics. Although an introductory course in Computer Graphics would be helpful, it is not a prerequisite for this course. Unlike the introductory graphics course currently offered under csci391, this course will use OpenGL and related APIs for most rendering. Topics may include GLSL (the OpenGL Shader Language), texture and bump mapping, mesh refinement, BRDFs, anti-aliasing algorithms, ray tracing, and radiosity. The course will include readings that will be discussed during class, a midterm and final, and a programming project meant to be a significant effort on a topic determined by the student. Some programming assignments may be given, especially at the beginning of the course. Although the course is listed as ENGR 691, it may be possible for advanced undergraduates to take it under a different course number as an independent study, with permission of the instructor.

**Engr 692-66: Special Topics in Engineering Science (Machine Learning)**

Instructor: Dr. Yixin Chen (ychen@cs.olemiss.edu)
Assistant Professor, Computer and Information Science

Time/Room: 2:30 TTh, Weir 106

Prerequisites:

Textbook: Pattern Recognition and Machine Learning, Christopher M Bishop, Springer, 2006.

Description:

This course seeks to introduce to the students the basic theory, concepts, and techniques of machine learning and give them a glimpse in the state-of-the-art of the area. Upon completion of this course, students will attain knowledge and skill of converting a machine learning algorithm discussed in the class to a computer program. The students should be able to formulate a real world application as a learning problem and applied proper machine learning techniques.

Topics to be covered include Bayesian decision theory, maximum-likelihood estimation, Bayesian estimation, nonparametric techniques, Linear discriminant analysis, computational learning theory, support vector machines and kernel methods, boosting, clustering, and dimensional reduction.