Summary:

The goal of this course is to read through Brian Kernighan and Rob Pike’s book *The Practice of Programming* (PoP) and thereby gain an understanding of techniques that make individual programmers more effective and productive.

Description:

Have you ever...

- wasted a lot of time coding the wrong algorithm?
- used a data structure that was much too complicated?
- tested a program but missed an obvious problem?
- spent a day looking for a bug you should have found in five minutes?
- needed a program to run three times faster and use less memory?
- struggled to move a program from a workstation to a PC or vice versa?
- tried to make a modest change to someone else’s program?
- rewritten a program because you couldn’t understand it?

If so you need to read PoP.

PoP is about much more than than writing code: it covers testing, debugging, portability, performance, and design alternatives—topics which are not usually the focus of university teaching.

Your task in this class will be to read and understand PoP. It is to the point (roughly 200 pages) and very well written. Each chapter includes examples, exercises, and further reading. Source for the examples is available online.

Text: The course will be based entirely on the following book:


Prerequisites:

All Option III Software Engineering graduate students should already have the programming background needed to appreciate PoP.

Web site:

All material related to the course is available at [www.ece.utexas.edu/~adnan/pop](http://www.ece.utexas.edu/~adnan/pop)
Format/Evaluation:

I will assign written homeworks consisting of exercises from PoP. The homeworks will not be programming intensive—their purpose is to make you read PoP in an organized manner.

Grades will be assigned solely on the basis of performance on the homework. I will grade the homeworks and return them by US mail.

Outline:

Chapter 1  Style—naming, comments, motivation.

Chapter 2  Algorithms and Data Structures—arrays, lists, trees, hashing; libraries.

Chapter 3  Design—Markov chain example, implementation in various languages.

Chapter 4  Interfaces—example, principles, UIs.

Chapter 5  Debugging—tools, clues, nonreproducible bugs.

Chapter 6  Testing—systematic testing, automation, scaffolding

Chapter 7  Performance—profiling, strategies, space efficiency.

Chapter 8  Portability—languages, libraries, organization.

Chapter 9  Notation—formatting data, programmable tools, interpreters, virtual machines.