Syllabus

About Perl

Perl is a powerful and unique interpreted language specifically suited for text processing (it is often said to be an acronym of Practical Extraction and Reporting Language, although this phrase came about well after Perl was designed and named). It was developed in 1987 by Larry Wall to provide greater flexibility for his own projects. It soon caught on and was used across the world for CGI applications and system administration tasks. It incorporated many features from various languages including dynamic typing, object, and first-class functions (closures). The versatility extends into the syntactic structure as well, allowing one to write code that is most natural for him/her. The parser is adept at resolving ambiguities and has lead to two popular phrases, "Do What I Mean" and the most common "There's More Than One Way To Do It."

Students will learn the basics of programming in Perl through writing weekly programs and readings about style and structure of the language. Obviously an in depth study of the parser is beyond the scope of this class, but where appropriate students will learn how the language is structured with a focus on processing text (using another keystone of Perl – regular expressions). The final project will involve writing a web crawler to traverse a local repository of web pages.

Week 1

Week one will focus on introducing students to Perl and getting them ready for many of the idiosyncrasies of Perl (for this reason printouts for each week will include key aspects in which Perl differs from other languages). This will include values that Perl considers to be false (there is no boolean data type), lack of an integer type, dynamic arrays, namespace, scalar vs. list contexts, and default variables.

Week 2

Week two will focus on subroutines (functions) and file handles. As with everything else Perl handles subroutines differently from most other languages but by the end of the week students will be writing their own subroutines using Perl's structure. Students will also learn different ways of reading input from a file and the advantages of each method.

Week 3

Week three will cover hashes (associative arrays) and a quick introduction to regular expressions (more on these next week). Students will learn about character classes and simple quantifiers before getting a chance to write some simple regular expressions

Week 4

Week four goes much more into depth on regular expressions and the engine that supports them. They will learn about option modifiers, text anchors, match variables, more quantifiers, precedence within a regular expression, and greediness.

Week 5

Week five will explain additional control structures, expression modifiers (a conditional following an expression), loop controls, and advanced sorting techniques. Don't worry, this is not a recap of bubble sort or merge sort (by default Perl implements merge sort for you); rather this allows you to specify the ordering without having to write your own sorting algorithm.

Week 6

Week six will give students a cursory overview of modules (although they will not need any of them, any Perl programming done after this course will likely benefit from including appropriate modules). Students will also learn about file tests and directory operations.

Week 7

Week seven will cover some advanced topics including error trapping and references. To those not familiar with pointers, references may be a little confusing at first but they are well worth the learning curve. Students will learn how to create complex data structures (including circular and anonymous structures) as well as how memory is managed internally using scope and reference counting.

Week 8

Week eight will involve reading several papers about strategies for crawling the web. Students will learn various methods for crawling the web as a search problem as well the importance of downloading high-quality pages early in the process (and what constitutes a "high-quality" page). Students will also learn about ethical crawling practices.

Final

The final project will involve writing a web crawler to crawl a local repository of web pages. This will avoid potential issues with network activity as the code is developed as well as allowing for a more controlled test environment. Students will not need to know any network programming but much of the practice of writing a real-world web crawler will be the same.