

Foundations of Data Processing in Library and Information Science

(LIS 317)

Section A Thursdays, 9–11:50 AM Room 133, Armory

David Dubin

Office: LIS 222

Office hours: Thursdays, 2–5 PM

Phone: 217–244–3275 (217–BIG–EARL)

E-mail: dubin@alexia.lis.uiuc.edu

Web: <http://www.lis.uiuc.edu/~dubin/>

This document is Copyright © 2000 by David Dubin and the Trustees of the University of Illinois. In addition to this syllabus, this course is governed by the rules and guidelines set forth in the document A Handbook for Graduate Students and Advisers which students receive upon admission to the program. Students should also consult, and take to heart, the Professional Guidelines and Codes of Ethics for Library and Information Science Professionals available from the GSLIS main office.

This syllabus is provided to UIUC students as part of the materials for a particular class. However, it may be copied, redistributed, and modified under the terms of the [OpenContent License](http://www.opencontent.org) (Version 1.0). The text of that license is available on the Worldwide Web at www.opencontent.org. Resources that are linked to or referenced from within this syllabus (e.g., readings, outlines, discussions) are not covered by the OpenContent License, unless specifically labeled as such.

REQUIRED TEXTS

Jon Orwant. *Perl 5 Interactive Course*, Certified Edition (Waite Group Press , 1998).

SCOPE AND OBJECTIVES

This course covers the common data processing constructs and programming concepts used in library and information science. The history, strengths and weaknesses of the techniques are evaluated in the context of our discipline. These constructs and techniques form the basis of applications in areas such as bibliographic records management, full text management and multimedia. No prior programming background is assumed.

Objectives

- Present an introduction to computer programming concepts and techniques, geared toward the needs of LIS students and professionals.
- Demonstrate tools to support challenging projects in other LIS classes, and problem solving on the job.

- Provide experience with Perl, a language well suited to text processing, rapid prototyping, and quick solutions to specific problems. Ultimately, Perl can be combined with other languages for more advanced development projects than will be covered in this class.

THIS SYLLABUS

The official syllabus for this course is the SGML version that is linked off the class web page. Expressions of the syllabus in other formats are derived from the SGML version. The current SGML version should be consulted to resolve any inconsistencies among other renditions.

ACCESSIBILITY

To insure that disability-related concerns are properly addressed from the beginning of class, students with disabilities who require reasonable accommodations to participate are asked to contact the instructor as early as possible.

HALF UNIT REGISTRATION

Students enrolled in the class for a half unit are expected to complete all assignments with the exception of the term project. Their semester grades will be based on the homework assignments and class participation.

BASIS FOR EVALUATION

Students are responsible for their performance in meeting their own educational goals and those of the course; instructors are responsible for providing guidance, expertise, and support to help students reach those goals. Students are expected to participate in class exercises and discussions. Satisfactory work will receive a grade in the C range, good work will receive a grade in the B range, and superior work will receive a grade in the A range.

Final grades for students enrolled for a full unit will be calculated as follows:

- First Programming Assignment: 15%
- Second Programming Assignment: 15%
- Third Programming Assignment: 15%
- Fourth Programming Assignment: 15%
- Term Project: 30%
- Class Participation: 10%

Final grades for students enrolled for a half unit will be calculated as follows:

- First Programming Assignment: 22%
- Second Programming Assignment: 22%
- Third Programming Assignment: 22%
- Fourth Programming Assignment: 24%
- Class Participation: 10%

On Adapting the Work of Others

Criteria for grading assignments include (but are not limited to) creativity and the amount of original work demonstrated in the assignment. However, students are permitted to use and adapt the work of others, provided that the following guidelines are followed:

- Use of other people's material must not infringe the copyright of the original author, nor violate the terms of any licensing agreement. Know and respect the principles of fair use with respect to copyrighted material.
- Students must scrupulously attribute the original source and author of whatever material has been adapted for the assignment. Summarize (e.g. using source code comments) the changes or adaptations that have been made. Make plain how much of the assignment represents original work.

Assignment 1: Variables Branching and Looping

This assignment is due (by email to the instructor) no later than September 28.

Write a Perl program and document it with appropriate comments. The program should demonstrate the use of loops, branching logic, and named variables. The exact functionality of the program is up to you. Examples of what you might program include the following:

- Use Perl's `localtime` function in a program to determine whether a library book is overdue, and compute a fine.
- Research the algorithm for computing an ISBN check digit, and write a program for validating an ISBN entered by the user. Descriptions of the algorithm can be found at the [International ISBN Agency](#), and [elsewhere on the web](#).

Assignment 2: Subroutines and File I/O

This program is due (in email) to the instructor no later than October 19.

Write a program and document it with appropriate comments. The program should have a main routine and at least two functions or subroutines. The program should read input from a disk file and write output to a disk file. What else the program does, exactly, is up to you. Use your imagination.

Assignment 3: Stack Demonstration

This program is due (in email) to the instructor no later than November 2.

Write a program and document it with appropriate comments. The program should use a stack to perform some interesting computation. For example, you can write a four-function RPN calculator with functionality and interface similar to the Unix `dc` utility ("man dc" for documentation). Your stack need not be a formal Perl object: you can use a simple Perl list as shown in class. But your stack ought to work like a stack (e.g. with push and pop operations).

Assignment 4: Object Class Design

This program is due (in email) to the instructor no later than November 16.

Design an original object class, like those we've discussed. Write an application program that implements and employs the class. Follow these guidelines:

- Decide in detail what attributes ought to be used to implement the class. Document these decisions using comments.
- Design at least two method functions (not including constructors, destructors, and copy methods) for the class. Demonstrate their use in the application.
- Implement the class as a Perl object (i.e. a module).

Term Project

All students enrolled in the class must complete a final project that has been approved by the instructor. The final project represents in-depth investigation of a topic related to data processing, in the form of a computer program and supporting documentation. Proposals for term projects are due no later than November 9. Completed projects are due on December 7.

The program must be submitted to the instructor as machine-readable source code, but may be programmed in any language approved by the instructor. Like the program, the documentation must be prepared and submitted in machine-readable form. It should either conform to a standard online format (such as POD or man), or take the form of a report prepared using a declarative markup language (e.g., L^AT_EX format).

Class Participation

The class participation grade is based on consistent attendance, contribution to in-class and/or online discussions, and providing assistance to classmates outside of class. Please alert the instructor if a classmate has been of help to you outside of class.

SEMESTER OUTLINE

Part I: **Introduction**

August 24

Unix utilities and file management

Readings: Syllabus

Part II: **Variables and Scalar data**

August 31

Characters and character encodings

Readings: Orwant chapter 1, lessons 1–4

Part III: **Strings, arrays, and control structures**

September 7

Stacks

Readings: Orwant chapter 1, lessons 5–8

Part IV: **Regular Expressions and finite state machines**

September 14

Models of computation

Readings: Orwant chapter 2

Part V: **File input and output**

September 21

Readings: Orwant chapter 3

Part VI: **Functions and Subroutines**

September 28

Readings: Orwant chapter 4

Assignment 1:
variables, branching,
and loops: Due at 5 PM.

Part VII: **Hash Tables and Matrices**

October 5

Computational complexity

Readings: Orwant chapter 5

Part VIII: **Drivers and Formatting**

October 12

Readings: Orwant, Chapter 6

Part IX: **Complex data structures**

October 19

References, anonymous storage

Readings: Orwant chapter 7, lessons 1–3

**Assignment 2:
Subroutines and File**

I/O: Due at 5 PM.

Part X: **Object Orientation Part 1**

October 26

Methods, Introduction to UML

Readings: Orwant chapter 7, lesson 4

Part XI: **Object Orientation Part 2**

November 2

Inheritance, polymorphism, UML part 2

Readings: Orwant chapter 7, lesson 5

Assignment 3: Stack

demonstration: Due at 5 PM.

Part XII: **Object Orientation Part 3**

November 9

Privacy, operator overloading

Readings: Orwant chapter 7, lessons 6–8

Term Project

proposal: Due at 5 PM.

Part XIII: **Modules**

November 16

Readings: Orwant, chapter 8

Assignment 4: Object

Class Design: Due at 5 PM.

Part XIV: **Thanksgiving Break**

November 23

Part XV: **Processes and IPC**

November 30

pipes, message queues, semaphores

Readings: Orwant, chapter 9

Part XVI: **Wrapup and Evaluation**

December 7

Term project: Due at 5 PM.