A LITA Guide

THE LIBRARIAN'S INTRODUCTION TO PROGRAMMING LANGUAGES

EDITED BY

Beth Thomsett-Scott

The Librarian's Introduction to Programming Languages

Library Information Technology Association (LITA) Guides

Marta Mestrovic Deyrup, Ph.D., Acquisitions Editor,
Library Information and Technology Association,
a Division of the American Library Association

The Library Information Technology Association (LITA) Guides provide information and guidance on topics related to cutting-edge technology for library and IT specialists.

Written by top professionals in the field of technology, the guides are sought after by librarians wishing to learn a new skill or to become current in today's best practices.

Each book in the series has been overseen editorially since conception by LITA and reviewed by LITA members with special expertise in the specialty area of the book.

Established in 1966, the Library and Information Technology Association is the division of the American Library Association (ALA) that provides its members and the library and information science community as a whole with a forum for discussion, an environment for learning, and a program for actions on the design, development, and implementation of automated and technological systems in the library and information science field.

Approximately twenty-five LITA Guides were published by Neal-Schuman and ALA between 2007 and 2015. Rowman & Littlefield took over publication of the series beginning in late 2015. Books in the series published by Rowman & Littlefield are:

Digitizing Flat Media: Principles and Practices
The Librarian's Introduction to Programming Languages

The Librarian's Introduction to Programming Languages

A LITA Guide

Edited by Beth Thomsett-Scott

ROWMAN & LITTLEFIELD
Lanham • Boulder • New York • London

Published by Rowman & Littlefield
A wholly owned subsidiary of The Rowman & Littlefield Publishing Group, Inc.
4501 Forbes Boulevard, Suite 200, Lanham, Maryland 20706
www.rowman.com

Unit A, Whitacre Mews, 26-34 Stannary Street, London SE11 4AB

British Library Cataloguing in Publication Information Available

Library of Congress Cataloging-in-Publication Data Available

ISBN 978-1-4422-6332-1 (cloth : alk. paper)
ISBN 978-1-4422-6333-8 (pbk : alk. paper)
ISBN 978-1-4422-6334-5 (ebook)

 The paper used in this publication meets the minimum requirements of American National Standard for Information Sciences—Permanence of Paper for Printed Library Materials, ANSI/NISO Z39.48-1992.

Printed in the United States of America

The Librarian's Introduction to Programming Languages is dedicated to all librarians seeking to take up programming and who feel that they don't have the basics to begin. It is my hope that this book will fill that void.

I also dedicate it to Carol Scott, my amazing daughter,
who perseveres through the highs and lows of life,
and David Scott, my best friend through the travels this world gives us.

Peace to all.

Contents

Illustrations

FIGURES

TABLES

EXAMPLES

Chapter 2: Python

Chapter 3: Ruby

Chapter 4: JavaScript

Chapter 5: Perl

Chapter 6: PHP

Chapter 7: SQL

Chapter 8: C

Chapter 9: C#

Acknowledgments

This book came to be through some circuitous events. The original editor needed to withdraw, and Marta Deyrup, editor for the LITA Guide series, put me in touch with Ron T. Brown, and we agreed to work as coeditors. After the original concept for the book was approved, Ron was offered a job in industry and accepted it. This new position kept Ron too busy to work further on the book. By this time, a few chapter authors had other commitments and I needed to recruit replacements. There was also a bit of delay while moving from a previous publisher to Rowman & Littlefield.

With the above said to set the stage, I want to offer my grateful thanks to Marta Deyrup for her confidence in me, Ron for guiding the concept of the book, and all my original authors who have hung in there for nearly two years. Many additional thanks to the amazing authors who stepped in at the near last minute to provide excellent chapters.

Preface

A Librarian's Introduction to Programming Languages provides an overview of the most popular programming languages used in libraries, including practical examples that will help librarians get past the "I can't do it" stage. We start with Python, Ruby, and JavaScript, which are considered the most readable and thus relatively easier-to-learn languages, which also have numerous applications. Perl, PHP, and SQL follow, as they are a bit trickier but incredibly useful. C, C#, and Java complete the book as compiled languages, with their own ins and outs, yet they have a number of important uses in libraries.

While there are books and websites devoted to teaching programming, few works address multiple programming languages or provide library-specific examples. The audience for this book is composed of administrators, practitioners, students, educators, and other lifelong learners interested in computer programming.

This book is designed to provide a basic working knowledge of each language presented, examples where the language is used in libraries, ideas of when a librarian would find each language useful, and small examples for practical experience. Recommended resources are provided for additional reading. The contributed-chapter format allows a wider spectrum of languages to be covered from a greater variety of experiences.

On behalf of the contributors, I wish all readers the very best of luck in creating new programs for their personal and professional use or consulting with programmers to implement a new or redesigned technology. We offer our sincere hope that these chapters guide you to a lifelong enjoyment of programming regardless of your current level of skill and knowledge. Happy programming!

Chapter One

Introduction

Dean Walton

HISTORY AND DEVELOPMENT

Programming has come a long way from its beginnings to today. My first introduction to programming was using punch tape, the same tape that was used to deliver changing stock market prices. For those familiar with the classic 1960s sitcom *The Adams Family*, Gomez was always pouring over his ticker tape. The next major developments were punch cards and Fortran (http://www.fortran.com/). With the advent of the personal computer, programming became accessible as a pastime for the masses. Programming is especially interesting since, in some cases, programmers may not even want or have to use a computer. A programmer can write a script on a piece of paper, comprehend the flow of logic, and never need to see it actually run. Programming is effectively the documentation of logic to solve a task. It can be a game, a puzzle to solve, just like the morning crossword. However, other people want to get a task done and use the computer to do it.

Computers are now integral to the work of librarians, and because many computer programs allow users to customize and run short program segments, called scripts, librarians could be missing out on a huge slice of the library world if they do not learn a few programming basics. As an example, if you have budget data, you can perform repetitive analyses to calculate various scenarios. With circulation data, you can perform repetitive analyses to examine use of items by certain populations, call number range use related to semesters, and other important relationships. Programming allows a person to perform complex repetitive tasks quickly and allows for quick modification as parameters change.

So how does a librarian with no programming experience start? First, let's understand that the terms *program* and *program code* are fairly interchange-

able. A program is a complete set of stand-alone program code. Program code typically refers to code in general, not a complete program. Programming is the action of creating a program, essentially writing the code that makes up a program. Code is the English-like instructions we humans use to tell a computer what to do. You may have heard of programmers or coders. These are essentially the same thing: people who write (or code) programs. One might say that learning to program is now kids' play. There are a plethora of books, online videos, and game-oriented websites that can help. One of these, Code .org, is a great site for basic instruction; it teaches the logic of programming regardless of the programming language. Code.org provides tutorials and content for elementary kids to learn the building blocks of programming. Of course, adults can learn from the site as well and do it in fun ways. If you want something more advanced, then YouTube is just a click away with its wealth of step-by-step videos. One must be careful, though, to ensure the quality of the instruction. The resources provided by chapter authors and the Additional Resources section of this book will guide you to quality items for your trek through the land of programming.

THE BASICS

It is important to make a distinction between interfacing with a computer and programming on it. A first step for many adults, including librarians, is to practice interfacing skills. A good way to start is to pull out an old desktop or laptop computer and load Linux on it. The newest versions of Ubuntu Linux, an operating system, will, when loaded, resemble Windows or Apple OS. However, in Linux, it is still possible to have a mouse-driven graphical user interface and enter the "Terminal" (X-terminal), where running the computer is command driven. By this I mean that the user can type commands on the appropriate lines on the computer, hence the name "command line." Gone is the mouse and its cursor. To move around the computer, the user uses the arrow keys. On a Windows machine, you can hold down the Windows key (often at the bottom left of the keyboard) while also holding down the X key. This brings up a menu for the command line and also the command line with administrator rights.

There are several important tasks to learn in the command line. In order to function in this environment, the user must learn how to move back and forth between different folders or directories; move, copy, or delete contents within these folders or directories; and find, download, and unpack programs from other sources. Mastering this suite of tasks will serve the librarian very well.

Once these are achieved, loading the sets of scripts, or libraries, needed for a particular programming language will be much easier.

The starting point for all of this is the path (figure 1.1). This is what allows the user to know where everything resides in the file structure of the operating system. The path follows terminology that is analogous to a tree. The structure that connects a tree to the earth and the trunk to the tree are the roots. The very core of the path is the *root directory*. The root structure contains the commands that allow the computer to function. Any deletion of files in the root and the computer won't work; thus the root directory is often protected. Protection in this case means needing a password to change parts of the root directory. If you are the administrator for the computer, that is, it's your computer or you are responsible for it, then you should know the root password. Creation of the root password normally happens when you load the operating system onto your computer.

The user can add or delete folders from the root directly just as a tree can grow new branches or a gardener can trim them away. But, remember, if you cut out the roots, the tree will die. A good question is how you know what branches your tree pathway has. You can type *dir* in the command line and

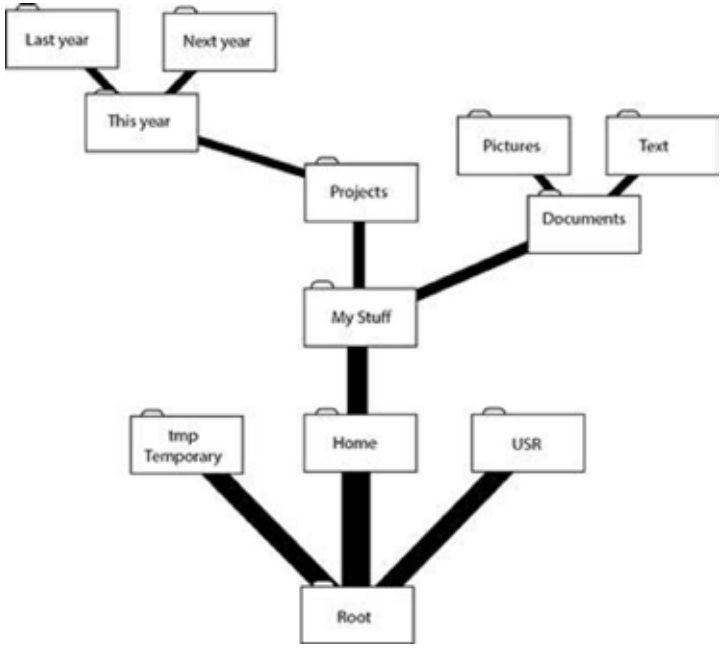

Figure 1.1. A generic path structure for folders or directories in a computer operating system.

see a list of folders contained within your location. To move into one of those folders, you type *cd*, short for "change directory," and the directory's name. There is even an easier way to go back to where you just were. By typing a single period on the command line and hitting enter, you ask the computer to bring you back to the previous folder. If you type two periods, you ask the computer to take you back to the starting point of all of the directories, back to the root directory.

There are also some added functions to many of these commands. A good example is how you can view the contents of a folder in Windows. You can view contents as a list or a list with details regarding size and creation date. The new user will be confronted with terms like *tar*, *grep*, and *bash*. Some of these refer to concepts that are easy to understand for someone used to Windows or Mac OS. For example, a .tar file is one that is compressed, like a zipped Windows folder. It needs to be uncompressed and the files contained within the folder moved to the appropriate folders or directories to be useful. Certain files need to be placed in an existing folder while others need to be placed in a brand-new folder that the user may need to create. It is important to read any associated text file or instructions on how to unpack and set up the new program.

It is likewise important to learn that Linux is designed to run some tasks without question while asking about others. This keeps the user from accidentally trashing the contents of the computer. In many Linux systems, this is defined as the *SUDO* user or super user. All that is really happening is that the operating system is asking the user to authenticate that she or he is the administrator of the computer and that the changes are actually desired. This helps to keep unauthorized users from making changes. In Linux, the user has the ability to peer into the operating system and pretty much make any changes that he or she requests, regardless of whether or not the change will stop the computer from functioning. This means the administrator has a lot of power, and a lot more power to mess up things than users of Windows have over their machines.

Once the user has Linux running on a computer and has decided which of the really cool programs to load for free, the next step is loading or downloading the packages for the language of your choice. The author has learned bits and pieces of programming based on what he wanted to do. In order to load and run the program BibApp, he had to learn the basics of a language called Ruby, but he didn't need to learn everything about Ruby. This is where learning a spoken language differs a great deal from learning a computer language. For program users, often only the very basics are needed to get a whole series of very different programs up and running. Actual programming, well, that is a different matter. Having a task always makes learning to program easier.