Introduction to Systems Programming:

a Hands-on Approach

by

Gustavo A. Junipero Rodriguez-Rivera and
Justin Ennen

Table of Contents

0. Introduction

Steve Jobs, one of the greatest visionaries of our time, once mentioned that a computer is like a "bicycle for the mind". He recalled that in an American Scientific magazine article he once saw, that they had created a ranking of the efficiency of locomotion in animals. The condor was the most efficient of all the animals with the least amount of energy needed to advance a kilometer. The humans were, not proudly, in the bottom third of the list. However, the ranking included also a "man on a bicycle" in the ranking. The man on a bicycle was in the top of the ranking. Steve Jobs concluded that computers are "like a bicycle for the mind".

Maybe the greatest asset that humans have compared to other animals is the mind. The mind has allowed us to overcome our limitations by inventing tools. We are now able to go faster, be stronger, and overcome our physical limitations more than in any other time in history thanks to the tools we invent. In the same way, the computers have allowed us to overcome some of the limitations of our minds. We are now able to do trillions of operations per second. We are now able to store more knowledge and have faster access to this knowledge. We are able to communicate to other humans faster and more efficiently than any other time in our history.

The use of computers does not come for free. It is necessary to write the software that makes all this magic possible. Without computer programs the computer would be just a pile of electronic components and wires. Software is what makes the computer come alive. The hardware is important, however the versatility of the computers come from the software that computer programmers write.

The large increase in productivity that we had in the last 20 years has been due in great part to the use of the computers and the Internet. The impact has been that manual jobs have been lost in exchange for jobs related to information technology. It is a paradox that we have a large unemployment rate while at the same time there are thousands of unfulfilled job openings related to information technology. It is more important than ever that we teach our younger generations to write the software of the future and to find new uses for computers. It is important to teach programming not only to computer majors but also biology, chemistry, medicine, education, physics and all other majors where computer applications will come from.

This book is targeted to students who have already learned how to program in C, have some basic programming skills, and would like to do the jump to write larger applications. Many books have been written on how to program. However, fewer books have been written on how to prepare mature programmers.

We want to give this book a practical approach about writing code. Dr. Shawn Ostermann, a friend and a coworker, once mentioned that there are three projects that a systems programmer has to write before graduation: a shell interpreter, a compiler, and an internet router. I agree with this assessment and also I would add to the list a web server and a memory allocator. I am sure

that this list of projects is biased and there are other projects that may be added to the list. However, with some certainty we can say that once a student completes these projects the student will be confident enough to attack much larger projects. It is the goal of this book to prepare the student to complete some of these projects.

Computer Science is now so complex that it is impossible to be an expert in every single layer of software. Starting from the silicon chip, computer architecture, assembly language, compilers, libraries, operating systems, graphical user interfaces, networking, databases, and ending with application programs. Not having knowledge on the multiple layers of software limits the problem solving capabilities and innovation.

For example, while trying to speedup the response of programs, not having knowledge of other layers of software that the application program depends on will make the optimization of the program difficult or almost impossible. Debugging problems is difficult if we see the other layers of software as just black boxes.

In this book we want to give enough knowledge of the different layers of software to make the optimization and debugging of programs easier. In addition, having knowledge of these layers of software will make innovations that span several of these layers possible.

This book was inspired by the material covered in the Systems Programming Course in the Computer Science Department at Purdue University. This is a sophomore level course that the students take after the C programming course. This course has been offered in the department for 5 years already with great success. Some knowledge of assembly language is recommended but not necessary.

We want to have a gentle introduction to Systems Programming that is understandable for most of the students. We want to try to bring aboard the majority of the students while at the same time offering some challenging projects to the students who want to go beyond the rest of the class. Our goal is to prepare the next generation of computer programmers and to help them see beyond the horizon of what our eyes can currently see.

Computer Science Department

Purdue University