Preface

To Everyone

Welcome to this book! We hope you'll enjoy reading it as much as we enjoyed writing it. The book is called **Operating Systems: Four Easy Pieces**, and the title is obviously an homage to one of the greatest sets of lecture notes ever created, by one Richard Feynman on the topic of Physics [F96]. While this book will undoubtedly fall short of the high standard set by that famous physicist, perhaps it will be good enough for you in your quest to understand what operating systems (and more generally, systems) are all about.

The four easy pieces refer to the four major thematic elements the book is organized around: **virtualization**, **concurrency**, **persistence**, and **distribution**. In discussing these concepts, we'll end up discussing most of the important things an operating system does, as well as a few things that distributed systems do. And hopefully, you'll also have some fun along the way. Learning new things is fun, right? At least, it should be.

Each major concept is divided into a set of chapters, most of which present a particular problem and then show how to solve it. The chapters are short, and try (as best as possible) to reference the source material where the ideas really came from. One of our goals in writing this book is to make the paths of history as clear as possible, as we think that helps a student understand what is, what was, and what will be more clearly. In this case, seeing how the sausage was made is nearly as important as understanding what the sausage is good for[1].

There are a couple devices we use throughout the book which are probably worth introducing here. The first is the **crux** of the problem. Anytime we are trying to solve a problem, we first try to state what the most important issue is; such a **crux of the problem** is explicitly called out in the text, and hopefully solved via the techniques, algorithms, and ideas presented in the rest of the text.

[1]Hint: eating! Or if you're a vegetarian, running away from.

We also use one of the oldest didactic methods, the **dialogue**, throughout the book, as a way of presenting some of the material in a different light. These are used to introduce the major thematic concepts (in a peachy way, as we will see), as well as to review material every now and then. They are also a chance to write in a more humorous style, which we greatly enjoy. Whether you enjoy it, well, that's another matter entirely.

At the beginning of each major section, we'll first present an **abstraction** that an operating system provides, and then work in subsequent chapters on the mechanisms, policies, and other support needed to provide the abstraction. Abstractions are fundamental to all aspects of Computer Science, so it is perhaps no surprise that they are also essential in operating systems.

Throughout the chapters, we try to use **real code** (not **pseudocode**) where possible, so for virtually all examples, you should be able to type them up yourself and run them. Running real code on real systems is the best way to learn about operating systems, so we encourage you to do as much of this as possible.

In various parts of the text, we have sprinkled in a few **homeworks** to ensure that you are understanding what is going on. These homeworks are usually little **simulations** of various pieces of an operating system; you should download the homeworks, and run them to quiz yourself. The homework simulators have the following feature: by giving them a different random seed, you can generate a virtually infinite set of problems; the simulators can also be told to solve the problems for you. Thus, you can test and re-test yourself until you have achieved a good level of understanding.

The most important addendum to this book is a set of **projects** in which you learn about how real systems work by designing, implementing, and testing your own code. All projects (as well as the code examples, mentioned above) are in the **C programming language** [KR88]; C is a simple and powerful language that underlies most operating systems, and is thus worth adding to your tool-chest of languages with which you are familiar. Two types of projects are available (ideas for which are provided in the appendix). The first type is based on what you would call **systems programming**; these projects are great for those who are new to C and Unix and want to learn how to do low-level C programming on such systems. The second type is based on a real operating system kernel developed at MIT called xv6 [CK+08]; these projects are great for students that already have some C background and really want to get their hands dirty inside the operating system. At Wisconsin, we've run the course in three different ways: either all systems programming, all xv6 programming, or a mix of both.

To Educators

If you are an instructor or professor who wishes to use these notes, please feel free to do so. As you may have noticed, they are free and available on-line from the following web page:

`http://www.cs.wisc.edu/~remzi/OSFEP`

At some point, you will be able to purchase a printed copy from a self-publishing site such as `lulu.com`. Look for it starting in Fall, 2011! Details will also be on the web page above.

The proper reference to the book is as follows:

> **Operating Systems: Four Easy Pieces**
> Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
> Version 0.4, September 2011 (or Version 0.3, if before that date)
> `http://www.cs.wisc.edu/~remzi/OSFEP`

The course divides fairly well across a 15-week semester, in which you can cover most of the topics within at a reasonable level of depth. Cramming the course into a quarter probably requires dropping one of the four pieces, likely the last piece on distribution. There are also a few chapters on virtual machine monitors, which we usually squeeze in sometime during the semester, either right at end of the large section on virtualization, or near the end as an aside.

One slightly unusual aspect of the book is that concurrency, a topic at the front of many OS books, is pushed off herein until the student has built an understanding of virtualization of the CPU and of memory. In our experience, students have a hard time understanding how the concurrency problem arises, or why they are trying to solve it, if they don't yet understand what an address space is, what a process is, or why context switches can occur at arbitrary points in time. Once they do understand these concepts, however, introducing the notion of threads and the problems that arise due to them becomes rather easy, or at least, easier.

You may have noticed there are no slides that go hand-in-hand with the notes. The major reason for this omission is that we believe in the most old-fashioned of teaching methods: chalk and a blackboard[2]. Thus, when we teach the course, we come to class with a few major ideas in mind and use the board to present them. In our experience, using slides encourages students to "check out" of lecture (and log into facebook.com), as they know the material is there (in powerpoint form) for them to digest later; using the blackboard makes lecture a "live" viewing experience and thus (hopefully) more interactive, dynamic, and enjoyable for the students in your class.

[2]We could say markers and a whiteboard, but that is less satisfying.

If you'd like a copy of the notes we use in preparation for class, please feel free to drop us an email. We hope to make a version of those class-preparation notes available online sometime.

To Students

If you are a student reading this book, thank you! It is a real honor for us to provide some material to help you in your pursuit of knowledge about operating systems. We both think back fondly towards some textbooks of our undergraduate days (e.g., Hennessy and Patterson [HP90]) and hope this book will become one of those good memories for you.

You may have noticed this book is free and available online. There is one major reason for this: textbooks are generally too expensive. This book, we hope, represents a new wave of free materials to help those in pursuit of their education, regardless of which part of the world they come from or how much they are willing to spend for a book.

We also hope, where possible, to point you to the original sources of much of the material in the book: the great papers and persons who have shaped the field of operating systems over the years. Ideas are not pulled out of the air; they come from smart and hard-working people (including numerous Turing-award winners[3]), and thus we should strive to celebrate those ideas and people where possible. In doing so, we hopefully can better understand the revolutions that have taken place, instead of writing texts as if those thoughts have been ever present [K62]. Further, perhaps such references will encourage you to dig deeper on your own.

[3]The Turing Award is the highest award in Computer Science; it is like the Nobel Prize, except that you have never heard of it.

Acknowledgments

This section will contain thanks to those who helped us put the book together. The important thing for now: **your name could go here!** But, you have to help. So send us some feedback and help debug this book. And you could be famous! Or, at least, have your name in some book.

Those who have helped so far include Kevin Liu, Natasha Eilbert, Cody Hanson, Matt Reichoff, Tuo Wang, Dan Sondergaard, Zef RosnBrick, Jay Lim, Alex Wyler, Ryland Herrick, Murugan Kandaswamy, Radford Smith, Mike Griepentrog, Ahmed Fikri, Dustin Metzler, Dustin Passofaro, Karl Wallinger, Srinivasan Thirunarayanan, Abhirami Senthilkumaran, Huanchen Zhang, Henry Abbey, Sy Jin Cheah, Cara Lauritzen, Brennan Payne, Ripudaman Singh, Nathan Sullivan, Lihao Wang, Balasubramanian, Meng Huang, Ross Aiken, Seth Pollen, Martha Ferris. To all of them: thanks! A special thanks to Kevin Liu (Fall '09), Ahmed Fikri (Spring '11), and Srinivasan Thirunarayanan (Fall '11), Abhirami Senthilkumaran (Fall '11), and Huanchen Zhang (Fall '11), whose fixes and comments went above and beyond the call of duty.

Many thanks to the hundreds students who have taken 537 over the years. In particular, the Fall '08 class who encouraged the first written form of these notes (they were sick of not having any kind of textbook to read – pushy students!).

A great debt of thanks is also owed to the brave few who took the xv6 project lab course, much of which is now incorporated into the main 537 course. From Spring '09: Justin Cherniak, Patrick Deline, Matt Czech, Tony Gregerson, Michael Griepentrog, Tyler Harter, Ryan Kroiss, Eric Radzikowski, Wesley Reardan, Rajiv Vaidyanathan, and Christopher Waclawik. From Fall '09: Nick Bearson, Aaron Brown, Alex Bird, David Capel, Keith Gould, Tom Grim, Jeffrey Hugo, Brandon Johnson, John Kjell, Boyan Li, James Loethen, Will McCardell, Ryan Szaroletta, Simon Tso, and Ben Yule. From Spring '10: Patrick Blesi, Aidan Dennis-Oehling, Paras Doshi, Jake Friedman, Benjamin Frisch, Evan Hanson, Pikkili Hemanth, Michael Jeung, Alex Langenfeld, Scott Rick, Mike Treffert, Garret Staus, Brennan Wall, Hans Werner, Soo-Young Yang, and Carlos Griffin (almost).

A final debt of gratitude is also owed to Aaron Brown, who first took this course many years ago (Spring '09), then took the xv6 lab course (Fall '09), and finally was a graduate teaching assistant for the course for the past two years or so. His tireless work has vastly improved the state of the projects (particularly those in xv6 land) and thus has helped better the learning experience for countless undergraduates and graduates here at Wisconsin. As Aaron would say in his usual succinct manner, "Thanks."

Final Words

Yeats famously said "Education is not the filling of a pail but the lighting of a fire." He was right but wrong at the same time[4]. You do have to "fill the pail" a bit, and these notes are certainly here to help with that part of your education; after all, when you go to interview at Google, and they ask you a trick question about how to use semaphores, it might be good to actually know what a semaphore is, right?

But Yeats's larger point is obviously on the mark: the real point of education is to get you interested in something, to learn something more about the subject matter on your own and not just what you have to digest to get an "A" in some class. As one of our fathers (Remzi's) used to say, "Learn beyond the classroom".

We created these notes to spark your interest in operating systems, to read more about the topic on your own, to talk to your professor about all the exciting research that is going on in the field, and even to get involved with that research. It is a great field(!), full of exciting and wonderful ideas that have shaped computing history in profound and important ways. And while we understand this fire won't light for all of you, we hope it does for many, or even a few. Because once that fire is lit, well, that is when you truly become capable of doing something great. And thus the real point of the educational process: to go forth, to study many topics, to learn and to mature, and most importantly, to find something that lights a fire for you.

Andrea and Remzi
Married couple
Professors of Computer Science at the University of Wisconsin
Chief Lighters of Fires, hopefully [5]

[4]If, that is, he actually said this; as with many famous quotes, the history of this gem is murky.

[5]If this sounds like we are admitting some past history as arsonists, you are probably missing the point. Probably.

References

[CK+08] "The xv6 Operating System"
Russ Cox, Frans Kaashoek, Robert Morris, Nickolai Zeldovich
From: http://pdos.csail.mit.edu/6.828/2008/index.html
xv6 was developed as a port of the original UNIX *version 6 and represents a beautiful, clean, and simple way to understand a modern operating system.*

[F96] "Six Easy Pieces: Essentials Of Physics Explained By Its Most Brilliant Teacher"
Richard P. Feynman
Basic Books, 1996
This book reprints the six easiest chapters of Feynman's Lectures on Physics, from 1963. If you like Physics, it is a fantastic read.

[HP90] "Computer Architecture a Quantitative Approach" (1st ed.)
David A. Patterson and John L. Hennessy
Morgan-Kaufman, 1990
A book that encouraged each of us at our undergraduate institutions to pursue graduate studies; we later both had the pleasure of working with Patterson, who greatly shaped the foundations of our research careers.

[KR88] "The C Programming Language"
Brian Kernighan and Dennis Ritchie
Prentice-Hall, April 1988
The C programming reference that everyone should have, by the people who invented the language.

[K62] "The Structure of Scientific Revolutions"
Thomas S. Kuhn
University of Chicago Press, 1962
A great and famous read about the fundamentals of the scientific process. Mop-up work, anomaly, crisis, and revolution. We are mostly destined to do mop-up work, alas.