Preface

This book aims to serve as a guide to the Xen hypervisor. The interface to paravirtualized guests is described in detail, along with some description of the internals of the hypervisor itself.

Any book about an open source project will, by nature, be less detailed than the code of the project that it attempts to describe. Anyone wishing to fully understand the Xen hypervisor will find no better source of authoritative information than the code itself. This book aims to provide a guided tour, indicating features of interest to help visitors find their way around the code. As with many travel books, it is to be hoped that readers will find it an informative read whether or not they visit the code.

Much of the focus of this book is on the kernel interfaces provided by Xen. Anyone wishing to write code that runs on the Xen hypervisor will find this material relevant, including userspace program developers wanting to take advantage of hypervisor-specific features.

Overview and Organization

This book is divided into three parts. The first two describe the hypervisor interfaces, while the last looks inside Xen itself.

Part I begins with a description of the history and current state of virtualization, including the conditions that caused Xen to be created, and an overview of the design decisions made by the developers of the hypervisor. The remainder of this part describes the core components of the virtual environment, which must be supported by any non-trivial guest kernel.

The second part focuses on device support for paravirtualized and paravirtualization-aware kernels. Xen provides an abstract interface to devices, built on some core communication systems provided by the hypervisor. Virtual equivalents of interrupts and DMA and the mechanism used for device discovery are all described in Part II, along with the interfaces used by specific device categories.

Part III takes a look at how the management tools interact with the hypervisor. It looks inside Xen to see how it handles scheduling of virtual machines, and how it uses CPU-specific features to support unmodified guests.

An appendix provides a quick reference for people wishing to port operating systems to run atop Xen.

Typographical Conventions

This book uses a number of different typefaces and other visual hints to describe different types of material.

Filenames, such as `/bin/sh`, are all shown in `this font`. This same convention is also used for structures which closely resemble a filesystem, such as paths in the XenStore.

Variable or function names, such as example(), used in text will be typeset like this. Registers, such as **EAX**, and instructions, such as **POP** will be shown in uppercase lettering. Single line listings will appear like this:

```
eg = example_function(arg1);
```

Longer listings will have line numbers down the left, and a gray background, as shown in Listing 1. In all listings, bold is used to indicate keywords, and italicized text represents strings and comments.

Listing 1: An example listing [from: example/hello.c]

```
#include <stdio.h>

int main(void)
{
	/* Print hello world */
	printf("Hello World!\n");
	return 0;
}
```

Listings which are taken from external files will retain the line numbers of the original file, allowing the referenced section to be found easily by the reader. The captions contain the original source in square brackets. Those beginning with `example/` are from the example sources. All others, unless otherwise specified, are from the Xen sources.

Comments from files in the Xen source code have been preserved, complete with errors. Since the Xen source code predominantly uses U.K. English for comments, and variable and function names, this convention has been preserved in examples from this book.

During the course of this book, a simple example kernel is constructed. The source code for this can be downloaded from:

http://www.prenhallprofessional.com/title/9780132349710.

Output from command-line interaction is shown in the following way:

```
$ gcc hello.c
$ ./a.out
Hello World!
```

A `$` prompt indicates commands that can be run as any user, while a `#` is used to indicate that root access is likely to be required.

Use as a Text

In addition to the traditional uses for hypervisors, Xen makes an excellent teaching tool. Early versions of Xen only supported paravirtualized guests, and newer ones continue to support these in addition to unmodified guests. The architecture exposed by the hypervisor to paravirtualized guests is very similar to x86, but differs in a number of ways. Driver support is considerably easier, with a single abstract device being exposed for each device category, for example. In spite of this, a number of things are very similar. A guest operating system must handle interrupts (or their virtual equivalent), manage page tables, schedule running tasks, etc.

This makes Xen an excellent platform for development of new operating systems. Unlike a number of simple emulated systems, a guest running atop Xen can achieve performance within 10% that of the native host. The simple device interfaces make it easy for Xen guests to support devices, without having to worry about the multitude of peripherals available for real machines.

The similarity to real hardware makes Xen an ideal platform for teaching operating systems concepts. Writing a simple kernel that runs atop Xen is a significantly easier task than writing one that runs on real hardware, and significantly more rewarding than writing one that runs in a simplified machine emulator.

An operating systems course should use this text in addition to a text on general operating systems principles to provide the platform-specific knowledge required for students to implement their own kernels.

Xen is also a good example of a successful, modern, microkernel (although it does more in kernelspace than many microkernels), making it a good example for contrasting with popular monolithic systems.

Acknowledgments

First, I have to thank Mark Taub for the opportunity to write this book. Since first contacting Mark in 2002, he has given me the opportunity to work on several projects. This included working with Mark Sobell, from whom I learned a lot about writing.

I also have to thank Debra Williams Cauley who coordinated everything for this book, along with the rest of her team who helped to transform it into the form you are now seeing.

I began writing this book near the end of the third year of my Ph.D., and would like to thank my supervisor, Professor Min Chen, for his forbearance when my thesis became a lower priority than getting this book finished. I would also like to thank the other members of the Swansea University Computer Science Department who kept me supplied with coffee while I was writing.

For technical assistance, I could have had no one more patient than Keir Fraser who answered my questions in great detail by email and in person when I visited XenSource. Without his help, this book would have taken a lot longer to write. A number of other people at XenSource and at the Spring 2007 XenSummit also provided valuable advice. I'd like to thank all of the people doing exciting things with Xen for helping to make this book so much fun to write.

I would also like to thank Glenn Tremblay of Marathon Technologies Corp. who performed a detailed technical review. While I can't guarantee that this book is error free, I can be very sure it wouldn't have been without his assistance. Glenn is a member of a growing group of people using Xen as a foundation for their own products, and I hope his colleagues find this book useful.

This book was written entirely in Vim. Subversion was used for revision tracking and the final manuscript was typeset using LaTeX. Without the work of Bram Moolenaar, Leslie Lamport, Donald Knuth, and many others, writing a book using Free Software would be much harder, if not impossible.

Finally, I would like to thank all of the members of the Slashdot community for helping me to procrastinate when I should have been writing.