Foreword

Over the past decade, a regrettable idea took hold: Operating systems, while interesting, were a finished, solved problem. The genesis of this idea is manifold, but the greatest contributing factor may simply be that operating systems were not understood; they were largely delivered not as transparent systems, but rather as proprietary black boxes, welded shut to even the merely curious. This is anathema to understanding; if something can't be taken apart—if its inner workings remain hidden—its intricacies can never be understood nor its engineering nuances appreciated. This is especially true of software systems, which can't even be taken apart in the traditional sense. Software is, despite the metaphors, information, not machine, and a closed software system is just about as resistant to understanding as an engineered system can be.

This was the state of Solaris circa 2000, and it was indeed not well understood. Its internals were publicly described only in arcane block comments or old USENIX papers, its behavior was opaque to existing tools, and its source code was cloistered in chambers unknown. Starting in 2000, this began to change (if slowly) —heralded in part by the first edition of the volume that you now hold in your hands: Jim Mauro and Richard McDougall's *Solaris™ Internals*. Jim and Richard had taken on an extraordinary challenge—to describe the inner workings of a system so complicated that no one person actually understands all of it. Over the course of working on their book, Jim and Richard presumably realized that no one book could contain it either. Despite scaling back their ambition to (for example) not include networking, the first edition of *Solaris™ Internals* still weighed in at over six hundred pages.

The publishing of *Solaris™ Internals* marked the beginning of change that accelerated through the first half of the decade, as the barriers to using and understanding Solaris were broken down. Solaris became free, its engineers began to talk about its implementation extensively through new media like blogs, and, most importantly, Solaris itself became open source in June 2005, becoming the first operating system to leap the chasm from proprietary to open. At the same time, the mechanics of Solaris became much more interesting as several revolutionary new technologies made their debut in Solaris 10. These technologies have swayed many a naysayer, and have proved that operating systems are alive after all. Furthermore, there are still hard, important problems to be solved.

If 2000 is viewed as the beginning of the changes in Solaris, 2005 may well be viewed as the end of the beginning. By the end of 2005, what was a seemingly finished, proprietary product had been transformed into an exciting, open source system, alive with potential and possibility. It is especially fitting that these changes are welcomed with this second edition of *Solaris™ Internals*. Faced with the impossible task of reflecting a half-decade of massive engineering change, Jim and Richard made an important decision—they enlisted the explicit help of the engineers that designed the subsystems and wrote the code. In several cases these engineers have wholly authored the chapter on their "baby." The result is a second edition that is both dramatically expanded and highly authoritative—and very much in keeping with the new Solaris zeitgeist of community development and authorship.

On a personal note, it has been rewarding to see Jim and Richard use DTrace, the technology that Mike Shapiro, Adam Leventhal, and I developed in Solaris 10. Mike, Adam, and I were all teaching assistants for our university operating systems course, and an unspoken goal of ours was to develop a pedagogical tool that would revolutionize the way that operating systems are taught. I therefore encourage you not just to read *Solaris™ Internals*, but to *download* Solaris, *run* it on your desktop or laptop or under a virtual machine, and *use* DTrace yourself to see the concepts that Jim and Richard describe—live, and on your own machine!

Be you student or professional, reading for a course, for work, or for curiosity, it is my pleasure to welcome you to your guides through the internals of Solaris. Enjoy your tour, and remember that Solaris is not a finished work, but rather a living, evolving technology. If you're interested in accelerating that evolution—or even if you just have questions on using or understanding Solaris—please join us in the many communities at `http://www.opensolaris.org`. Welcome!

Bryan Cantrill
San Francisco, California
June 2006