Introducing SeaOS

Daniel Bittman*, D J Capelis†, Darrell Long‡
*dbittman@ucsc.edu, †mail@capelis.dj, ‡darrell@soe.ucsc.edu
Working-group on Applied Security & Privacy
University of California, Santa Cruz
Santa Cruz, Ca, USA

***Abstract*—SeaOS is a simple operating system developed for fast prototyping of research concepts. SeaOS implements a POSIX-like interface in less than 30,000 lines of C kernel code. Its userspace is complete enough to be self-hosting and includes ports of gcc, bash, grub, binutils, and coreutils. SeaOS runs on modern x86 and x86_64 architectures and is designed to support our lab's operating system research in scheduling, virtualization and security architectures.**

I. Introduction

In recent years, the emergence of new operating system kernels has grown less frequent. Modern computing focuses effort on a few operating system kernels and has expanded these kernels to support an increasingly wide breadth of use-cases and workloads. While focus on a few operating system kernels has simplified much of the computing landscape, the complexity of a modern operating system has become an increasingly difficult environment to prototype the design of new fundamental systems concepts. With new areas such as virtualization and multi-chip architectures and continuing challenges such as the increasing importance of the storage hierarchy and energy in determining the bounds of system performance, continued innovation of core operating system concepts remains essential.

SeaOS is under active development in our lab to provide an operating system kernel which solves this problem. SeaOS provides researchers with a simple, straightforward codebase that is easy to modify, runs on modern hardware and can quickly be re-written to explore new concepts. The design of the kernel is organized around the principles of simplicity, modularity and feature minimalism. While SeaOS aims towards complete functionality to support prototyping advanced operating systems concepts, it attempts to find the minimal number of features necessary to implement this functionality.

II. Features

In support of our research goals for SeaOS, the areas the kernel focuses its feature set are architecture support and storage. These are the core areas of focus for our research and developing features in these areas enables us to use SeaOS as a prototype system. In addition, SeaOS also focuses on areas needed for rapid prototyping and extendability, including modularity and abstraction.

A. Architecture Support

For developing research prototypes, it is important to support multiple architectures. A research system which only runs on x86_64 processors won't necessarily apply to mobile processors. Likewise, a research system that only runs on ARM won't necessarily translate to datacenter computing. SeaOS currently supports x86 and x86_64 architectures, with ports to SPARC and ARM (for mobile experiments) planned in the future.

SeaOS delineates the architecture specific code from the architecture dependent code to allow quick porting. Ports to new architectures can be accomplished by writing the architecture specific functions to implement SeaOS's feature set into a new architecture specific directory and recompiling the kernel for the new architecture.

In addition to ports to multiple architectures, new systems concepts require support for various architecture features. SeaOS already supports multiprocessing, multithreading and multicore architectures. But we've also found our research will require support for everything from performance counters and performance events to virtualization extensions. Implementing architecture features is a key area for allowing systems prototypes.

B. Storage

Providing a realistic model for how modern computing systems move data from one portion of the system to another and store it is critical for modern system performance. To provide a realistic system environment for research concepts to be tested and developed, SeaOS needs to implement a real storage stack. It is important that the storage stack not only work with real devices, but act in the same ways a real storage stack would act.

To this end, SeaOS supports ATA and AHCI devices which allow a wide range of storage devices to be used. The transfer mode from both is DMA, which is required in a modern storage stack and to store files, the operating system uses Linux's ext2 [1] with support for FAT and iso9660 currently underway.

C. Modularity

To aid in rapid prototyping and fast code development, SeaOS is designed to be modular. For example, the data structures for process queues and the implementation of the scheduler allow for the scheduling policy to be changed very easily. The kernel level implementation of malloc (for kernel objects) is designed to be able to have multiple allocators of different types active (in different memory regions) at any one

978-1-4799-4441-5/14/$31.00 ©2014 IEEE

time, and allows for the selection of which allocators to use at compile time. Each driver and component can be split into a module and loaded by the kernel as needed. Like other modular operating systems, SeaOS uses named symbols to resolve code locations.

D. Abstraction

One of the final principles behind the SeaOS kernel is abstraction, which attempts to ensure that all higher and complex features are built from a minimal set of simpler building blocks. It allows for multiple layers of simple abstraction that are well defined so to provide simpler and more readable kernel code.

For instance, for ease of porting all code that is architecture specific is separate, and provides only the minimal functionality for that architecture (paging code, context switching, etc). The remaining portions of the kernel are platform independent. For ease of adding filesystems, filesystem operations go through a virtual filesystem layer. For ease of support for various pieces of hardware, subsystems are written towards a generic hardware layer and drivers map those functions onto the specific hardware.

In a big way, this ties in the principle of simplicity. As an example of abstraction, the C library that the Sea userspace tools use is a library known as Newlib. [2] Newlib was designed for embedded systems, and so it is quite light-weight and simple. It is also very portable, and was very easy to get running on Sea. For most of its functionality, it only requires 17 system calls to be implemented, all of which are very basic Unix system calls (such as open and fork). A number of other system calls are implemented in order to extend the functionality of the C library, as required by the ported userspace programs.

III. Research Aims

While SeaOS has broad applicability as a research kernel for a wide variety of systems experiments and prototypes, our current efforts are focused around implementing the features needed to support two specific research projects within our lab: the *LockBox* project and our I/O Scheduling research.

A. The LockBox Project

The *LockBox* project, is an experiment to implement secure memory segments which allow applications to keep data secure from potentially malicious operating systems. We intend to prototype our concepts in SeaOS before eventually implementing our system in Linux. We plan to use SeaOS to aid our research in two distinct ways. The first is providing a simple test operating system to run on our custom hardware which can easily test and check the functionality of our system. Since our hardware platform is based on an OpenSPARC CPU core running an FPGA, we'll be porting SeaOS to SPARC.

The second way SeaOS will be used within the Lockbox project is to provide a pure software only system. SeaOS will gain support for VT-x and VT-d x86_64 intel virtualization extensions and become a hypervisor. From this hypervisor base, we'll have a simple system to emulate hardware features that do not currently exist and run other instances of SeaOS, or instances of other operating systems, like Linux, on this emulated platform.

B. I/O Scheduling Experiments

Our lab is currently conducting experiments around a new scheduling design for multicore architectures. The ongoing struggle to overcoming I/O bottlenecks which increasingly define the limitations of performance in modern computing systems may revolve around modifying the scheduler. SeaOS's existing simple scheduler has been designed to do simple round robin scheduling of active tasks.

We are implementing several experimental scheduling concepts in SeaOS, including a range to provide comparison to our new algorithms. For our research, we're also implementing support for performance counters, which provides CPU-specific profiling information. These counters will allow us to make scheduling decisions in SeaOS in a process specific context. Using this information, we hope to develop a scheduler which can use information about task I/O patterns to enhance system performance and responsiveness.

IV. Userspace

Unlike some systems, SeaOS uses its own tty code and terminal settings with working ports of *termcap*, *readline* and *ncurses* available to support compiling applications for SeaOS. This provides SeaOS with a fully capable terminal and supports a wide range of text-based interfaces and tools.

SeaOS's existing userspace is prepared for development, including the standard suite of *bash*, *coreutils*, *binutils*, *autotools*, *gcc* and everything needed to bootstrap a complete build environment. Newlib provides a standard C library. System maintenance is provided by a custom set of utilities, packaged as seaosutils, which implement essential operating system functions, like mount and shutdown.

The userspace is fully functional for the purposes of operating system development and continues to expand as the development team uses SeaOS for more tasks. We've found this is sufficient for most research purposes, though we eventually hope to implement minimal graphics support to allow the use of some graphical userspace tools, like a modern web browser, to enhance our ability to test and evaluate research concepts on a broader range of userspace tools.

V. Availability

The code of SeaOS is available on github. The kernel components can be found at: http://github.com/dbittman/seakernel and the userspace components can be found at: http://github.com/dbittman/sea.

VI. Conclusions

SeaOS is a small operating system which provides a rapid prototyping environment to support research into advanced operating system concepts. SeaOS is being actively used as a test bed in our lab to support our research. The support for

various platforms and their architecture features, as well as a fully working storage stack, are key features in supporting our research. The simplicity and modular design of the kernel allows researchers to quickly run experiments on new system concepts and prototype new concepts. Implementing research prototypes in a working operating system designed for rapid prototyping lowers the cost to change designs, experiment and innovate.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1018928.

REFERENCES

[1] R. Card and T. Tsó and S. Tweedie, *Design and Implementation of the Second Extended Filesystem*, *Proceedings of the First Dutch International Symposium on Linux*, 1994.

[2] C. Vinschen and J. Johnston, *The Newlib Homepage*, Available: http://sourceware.org/newlib/