SOSP Tutorial: Tock Operating System

http://www.tockos.org

Amit Levy[1], Daniel B. Giffin[1], Bradford Campbell[1], Paul Thomas Crews[1], Mateo Garcia[1], Branden Ghena[2], Shane Leonard[1], Pat Pannuto[2], Hubert Teo[1], Prabal Dutta[3], and Philip Levis[1]
{levya,dbg,pal}@cs.stanford.edu,{bradjc,brghena,ppannuto}@umich.edu,{ptcrews,mateog,shanel,hteo}@stanford.edu,
prabal@cs.berkeley.edu
[1]Stanford University, [2]University of Michigan, [3]UC Berkeley

ABSTRACT

The Tock operating system is a secure, embedded kernel for embedded and Internet of Things systems. Written in the Rust language, it supports kernel extensions in Rust as well as multiple concurrent applications written in C, Rust, or Lua. This tutorial will give an overview of the kernel's architecture and kernel programming in Rust. Attendees will write a kernel extension in Rust as well as a user-land networking application in C. Attendees will be provided hardware kits, which they may optionally purchase.

ACM Reference format:
Amit Levy[1], Daniel B. Giffin[1], Bradford Campbell[1], Paul Thomas Crews[1], Mateo Garcia[1], Branden Ghena[2], Shane Leonard[1], Pat Pannuto[2], Hubert Teo[1], Prabal Dutta[3], and Philip Levis[1]. 2017. SOSP Tutorial: Tock Operating System http://www.tockos.org. In *Proceedings of ACM Symposium on Operating Systems Principles, Shanghai, China, October 2017 (SOSP'17),* 1 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 LEARNING GOALS

The purpose of this tutorial is to give attendees an opportunity to

(1) learn about how Rust can be used in kernel programming,
(2) have a hands-on experience writing Rust code for an embedded kernel,
(3) learn about the Tock operating system as well as how it uses Rust to provide safety, and
(4) come away with hardware and the ability to do research on the platform.

2 MATERIAL OVERVIEW AND TARGET AUDIENCE

Tock is a secure embedded operating system for low-memory, low-energy microcontrollers, targeted at IoT and sensor network applications. Tock is unique among embedded operating systems because it provides isolation guarantees for both applications and the kernel. This enables highly reliable systems that incorporate potentially buggy or even untrusted components (e.g. end-user installed applications). Moreover, Tock provides many convenient mechanisms familiar from traditional operating systems such as preemptive scheduling, a portable system call interface, and dynamically loadable applications. Finally, The Tock kernel is written in Rust, a new type-safe systems language and provides, providing a instructive example of the techniques required to write resource-sensitive systems in such languages.

An in-depth report of the system goals and design will be presented at this year's SOSP.

The tutorial provides a hands-on introduction to the operating system. By the end, participants will have practiced writing a new kernel component as well as an end-to-end application in userspace and run both on real hardware. The tutorial is particularly appropriate for PhD students and researchers interested in new programming languages for operating systems kernels, systems programming in Rust, security, embedded systems, or Internet of Things applications.

Participants will learn the details, hands on, of an operating system that provides a secure foundation for low-memory systems. Tock provides a platform for future research in resource-constrained operating systems, secure IoT applications, and applying type-safe languages to systems software.

3 STRUCTURE

The tutorial has 3 major parts:

(1) Getting up and running with Tock. This part consists of 30 minutes compiling a Tock kernel and existing applications as well as a quiz whose answers require learning about part of the kernel code (to help with the next part).
(2) A 30 minute presentation on Tock's architecture and programming model, followed by writing a new capsule (the Tock term for a kernel module) in Rust. The capsule samples a light sensor and prints the readings to the kernel console. After completing this part, attendees will understand how they can extend the kernel to add new services or abstractions. It also gives a bit of experience in writing Rust code.
(3) Writing a new userspace application in C. This shows how the system call interface works, as well as how Tock handles dynamic memory allocation in the kernel (the subject of the SOSP paper). This shows how one can build new applications on top of Tock as part of a larger IoT system.

SOSP'17, Shanghai, China

DOI: 10.1145/nnnnnnn.nnnnnnn