Keerthana Kumar
kk8
Andrew Sharp
ags799

FINAL PROJECT - USER-LEVEL VIRTUALIZATION

Our final project was the creation of a library to allow a user program to create “threads.” These “threads” are other user programs, like hello.c. This library essentially would have thread management and creation functions like thread_create(), thread_run() and thread_destroy().

We began by implementing this functionality as a complete layer of abstraction between the thread and the host user program. We followed JOS’ lead in this effort by closely mimicking what occurs in kern/env.c in our own lib/thread.c. The first time that a user program asks to create a thread, a thrds array is initialized, similar to the envs array. The host user program would create a page directory for the thread to allow for another layer of memory abstraction on top of virtual memory. Next, it would load the binary of this thread similar to load_icode(), via a system call. The system call would place the binary into the address space of the host user program such that it appeared in the code segment of the thread. The thread would then run on the host user program, which would behave as a complete x86 emulator.

It quickly became apparent that such a task was far too time-consuming for our allotted time period. After discussion with the teaching staff we changed our plan to lessen the load on the user program, such that it would no longer have to emulate so much of the kernel and of the hardware. Firstly, the thread-creation functionality would be limited to the terminal, such that the user could enter “thread_create user_hello” to run hello.c. Also, we would use fdopen() to access the thread’s binary. This binary would have been compiled as position-independent code to support this action; we would have edited the Makefiles to allow for this. We would use the structure of the ELF header to get to the code and memmove() it to a section of memory obtained via malloc(). thread_run() would have been the command to begin execution of the thread.

Once execution of the thread began, the thread would continue to run until completion. Upon exit control would be returned to the host terminal.