Future Work

Most of the rudimentary designing and developing work of this project has been done. However, due to the time limits (I've been blocked on some problems and bugs for quite a long time) and my capability. There are still lots of work that haven't been done yet.

Still working on it, might be finished soon

- The current platform info collection in the kernel booting stage is hardcoded as it's not quite important, but we can make it be configured by the user.
- The current implementation only supports running the roottask at the moment, should be extended to run multiple seL4 applications.
- The current implementation has only been tested on simple applications, should be tested on more complicated applications. (e.g. capdl-loader, seL4 test suites)
- About emulating scheduling, I've just come out with the idea recently, still implementing it.
  To emulate the round robin scheduling in master kernel, we can use blocking sockets, so each time kernel uses the scheduling algorithm to determine the next running seL4 thread and only replies to it. Hence all other seL4 applications will block on the socket except one that has been chosen.
  To emulate the preemption, we can set a timer in the kernel emulator, and signal our seL4 application, then the signal handler routine is invoked and block on the socket until the next IPC message is passed from the kernel emulator telling it to resume.