Yalnix Project

Wes Robbins, Joseph Icopini, Austin Hull, Christine Johnson

Introduction to Yalnix

- Virtualization of DCS 58 Computer
- We are tasked with implementing basic operating system functionality
    - Virtual Memory
    - Kernel/Userland separation
    - Reading and running user programs

Our Goals

Our original goal was to complete up to checkpoint 4 of the yalnix project.

- Checkpoint 1 : Pseudocode
- Checkpoint 2: Kernel boots and runs idle
- Checkpoint 3: Init Process
- Checkpoint 4: Functioning Syscalls Fork, Exec, Wait

What we accomplished

- Checkpoint 1 : Pseudocode
- Checkpoint 2: Kernel boots and runs idle, TRAP_CLOCK and TRAP_KERNEL are functional
- Checkpoint 3: Init Process successfully runs, syscalls implemented include GetPid, Delay, Exit, Brk
- Checkpoint 4: All trap handlers implemented or refer to the undefined trap handler. The Fork and Exec syscalls are partially implemented

Overall Functionality

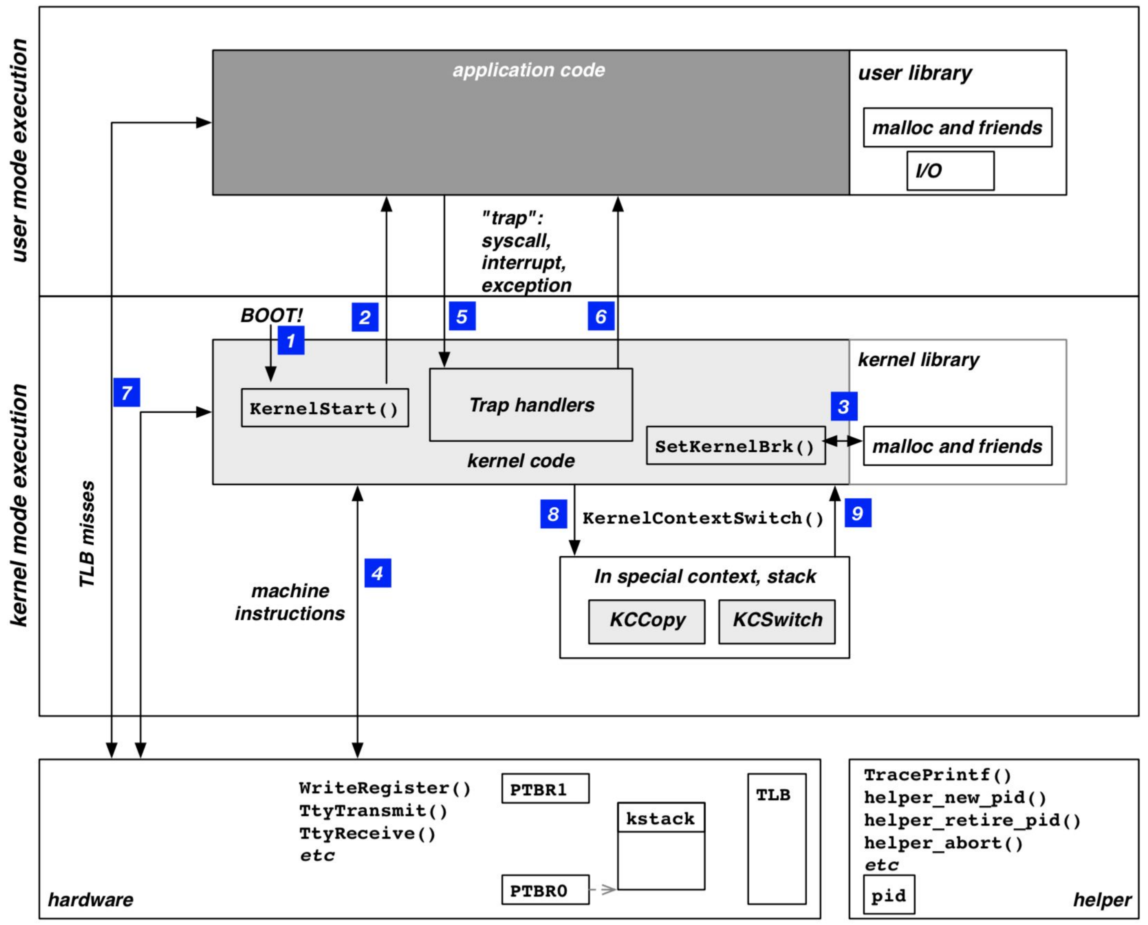


Figure 1.1: The overall flow of Yalnix.

Difficulties

- Working within Yalnix Framework
- Working with template files
- Working with Virtual Memory
- Using workarounds for functionality that wasn’t implemented yet

What we learned

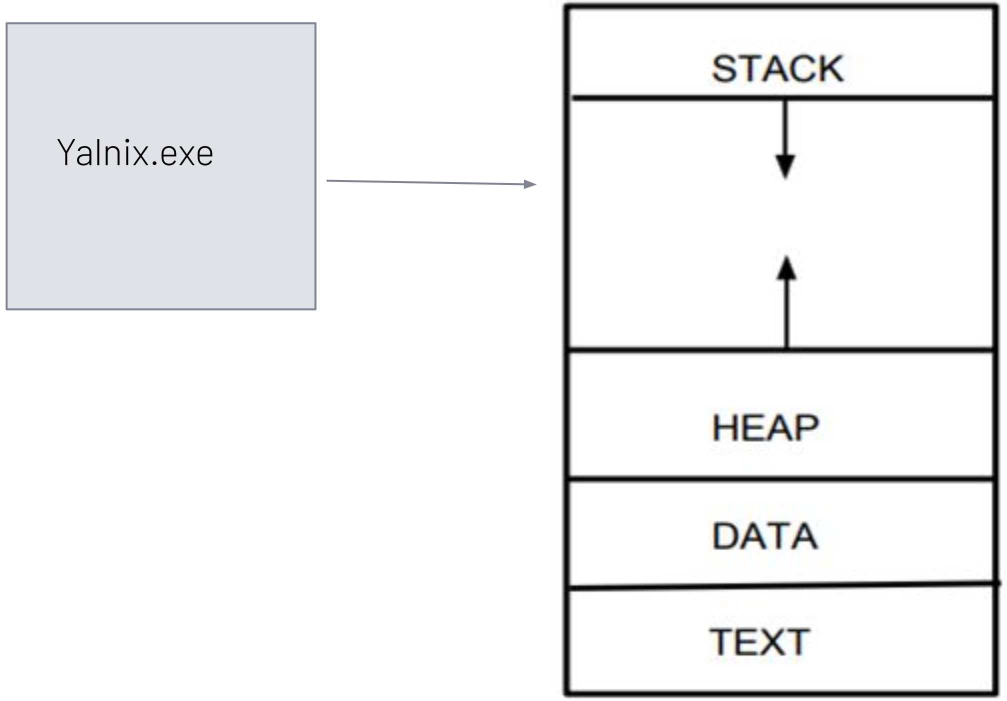
- A stronger technical understand of operating systems
- Gained experience in working on collaborative code
- File Organization

Demo Preview

Yalnix executable loaded in to page table

User executable loaded into yalnix

User program calls system calls to access kernel
-getpid()
-brk()
-fork()



Onto the Demo!