FYP Idea

Combine the resources of two separate machines into one. Each individual user-land process will be executed on one machine but it can access the resources of either machines.

Example: Since this is a microkernel, the filesystem is in userspace. All the drives of all the machines will be visible to the filesystem. When the filesystem tries to access a drive, it sends the request to the kernel. If this drive is local the kernel can perform the action as normal. If the drive is remote, the kernel can forward the request to the other kernel.

The main "in-depth" part of the project is designing the algorithm for choosing which process should run where. Accessing resources from a different kernel is likely to significantly slower than if the resource was local. So processes should be run on the machine that they use the most resources from. Problem: this isn't known before the program starts accessing resources. It may be possible to transfer execution of a process from one machine to another, but again this will take some time (have to copy memory).

Project Rough Plan

- Download and compile Redox (an existing microkernel OS written in Rust)
- Create a communication link between the two (or more) kernels
- Produce a distributed queue
- Change the process queue to this distributed queue instead. The kernel will probably only take own processes at this point.
- Change resource mappings so there are no conflicts (e.g. kernel A has eth0 and kernel B has eth1).
- Make it so if kernel A needs to access eth1, it uses the communication link
- Each kernel can take any *new* process from the queue. The process should stay on the same kernel at this point(so that memory doesn't need to be copied between machines)
- Allow an already started process to change kernel by copying the processes memory.

Extensions: To simplify the project I will initially use two kernels which are connected together on a LAN. I'll assume that both kernels remain online, and the communication be relatively instantaneous. However, if the project needs extending, I could increase the number of running kernels and I could also look into how to resolve when a kernel disconnects unexpectedly.