Memory Management Setup Plan

This document describes the memory management setup plan for Rusty OS.

1. During the boot loading process, stage 2 and stage 3 will create and install a temporary GDT so that stage 2 and stage 3 as well as when execution is handed over to the kernel may use memory greater than 1MiB. A temporary virtual memory system will also be setup so that the kernel can be loaded at a low physical address, say 1MiB, but it may think that it is executing in high memory, say 3GiB.
2. When execution is handed over to the kernel, the kernel boot up code will setup a new stack in a safe area of memory. This has to be done before the kernel actually begins to setup the operating system environment.
3. After setting up the stack the kernel must install its own GDT. The new GDT will be stored in the kernel’s global memory.
4. When the kernel begins execution and begins setting up the operating system environment, it must first setup a physical memory manager. The physical memory manager will be placed directly after the kernel image in memory. The physical memory manager will be designed to expand into higher memory so placing it directly after the kernel image is perfectly safe. The amount of memory it requires to manage all of the installed physical memory will be allocated during setup (i.e. the physical memory manager’s size will be static instead of dynamic). The physical memory manager will know how large the kernel image is and it will also know how large itself is and it will mark those memory regions used.
5. After the physical memory manager is setup the virtual memory manager needs to be setup. The master table will be setup in memory directly after the physical memory manager. The 3GiB page table will be stored directly after the master table but will only map enough addresses to cover the kernel image, physical memory manager and both virtual memory tables. The page table will be registered to the virtual memory manager just like any other page table. This will allow the rest of the table that isn’t immediately used by the kernel to be usable by any future process. This will allow the kernel to switch over to its own virtual memory manager.
   1. When the kernel or a process requests a block of 4kib memory, first the physical memory manager will find the first available free block using any appropriate means. It than tells the virtual memory manager which block of physical memory needs to be allocated and which block of virtual memory it needs to respond to. The virtual memory manager will than check the master page table to see if a corresponding page table already exists.
      1. If one already exists, the virtual memory manager will look up the page table in an internal list of tables. This page table address must be a virtual address that points to the physical address where the actual page table exists. It than updates the corresponding entry in the page table and refreshes the TLB.
      2. If one does not already exist, it must find a free virtual memory address block and assign a free physical memory block to it to be used for a new page table. After creating the new page table it must be registered with the internal list of page tables in the virtual memory manager. Than it proceeds with the first step.