Introduction:

This book will be about getting an OS up and running from scratch. I'm just in the planning phase, but here's the idea.

The book is open source. If you have a better way of doing something and can explain it in a clean manner then please contribute. The goal is to create an easy to read and understand manuscript that's 300 pages or less and will end with the reader having a functional if minimal OS. A hard rule for the first book is that each key topic below must be written in 20 pages or less! ←(no stealing pages from other topics). Boiler plate code can be shown as a github link if a written description of what to do takes less time to explain.

Book 1
1) Create a Cross Compiler GCC
2) Hello World Kernel
3) VGA Text Driver tty
4) GDT
5) Paging
6) Memory Manager
7) 64 bit mode
8) LIDT
9) ISR
10) IRQ
11) Exceptions
12) Inb Outb PortMappedIO
13) Brown's list of ports
14) Keyboard
15) Mouse
16) Hard drive
17) LibK and LibC

Book 2
18) Vesa mode
19) GUI – Draw something
20) Video Card Drivers (VCD)
21) Intel VCD
22) Nvidea (VCD)
23) Window controls
24) Text Editor

Book 3
25) Porting a program
26) Porting gcc
27) Internet

Book 4 …
28) USB
29) More Drivers
30) porting SDL
31) porting quake

- Attribution:
  - Michael - https://www.youtube.com/channel/UCaKToiAPfhOgxqy7Wuyjkqw
  - Andreas Kling - http://serenityos.org/
  - Eric Missimer - https://github.com/missimer
  - Erik Helin, Adam Renberg - https://littleosbook.github.io/book.pdf
  - OSDev Community - https://wiki.osdev.org/
  - Broken Thorn Entertainment Co. - http://www.brokenthorn.com
  -

Cross Compiler
A cross compiler is a compiler that has been built to run on the current system, but that outputs machine code that can run on a different OS (Or even on a different platform). For our purposes we want to create code that doesn't require any connection to the system that we are starting from. In this step we have several decisions to make. First is what we want our operating system to be named. This will be part of the "target triple" that we tell the cross compiler to build for. Another decision is whether we want to create 32 bit or 64 bit code. Right now we will be compiling for 32 bit code, see chapter 7 for the switch to 64 bit mode.

Hello World Kernel
By the end of this chapter you will have a tiny kernel that outputs the words "Hello World" on the screen and then halts. This is a powerful point in which your code and only your code is running on your computer. It is a high point of pride. We will be utilizing some assembly code here.

VGA Text Driver (tty)
By the end of this chapter you will have written your first driver which prints text to the screen in a controlled manner. Very useful for verifying that things are working properly.

GDT Global Descriptor Table
By the end of this chapter you will have a functional GDT. The GDT is used to determine what level of privilege code is allowed to run at and therefore access. The GDT is mainly set once and then operates in the background on a 64 bit system. Many of the features that it served in 16 and 32 bit modes have become obsolete with the advent of memory paging. It is still necessary to set an operational GDT to run before we can set up paging, interrupts, and other.

Paging
By the end of this chapter you will have a paging structure set in place.

Memory Manager
Being able to allocate memory dynamically is hugely important for programs running on your system. In order for these programs to run properly it is important to prevent memory allotments from overlapping.