# BareMetal Design Doc

## Introduction

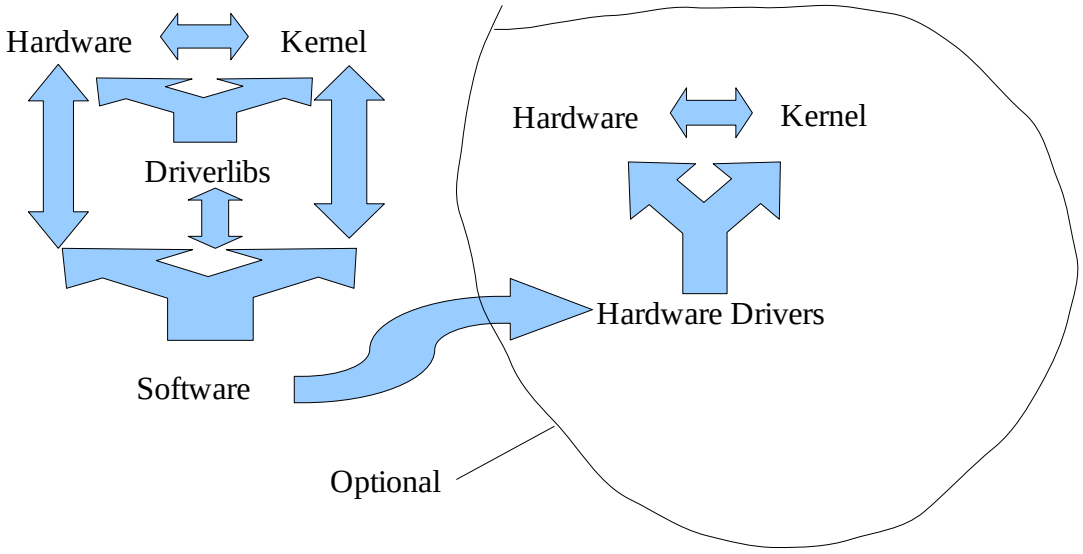
This is the design specification for the BareMetal Operating System. The first part of this design document will be a general overview of how the operating system works, the later part will be a detailed description of how specific things work like memory allocation and deallocation and such. BareMetal gets its name from being a very low level operating system and giving more control over things to the programmer. The Operating System does not aim to have POSIX compliance, instead it aims to create a completely original design for its kernel. BareMetal is meant not for the General Purpose user, but for advanced programmers who wish to test out their ideas under a stable OS while still having control over everything they need. Device drivers will be available, but will not be required to be used by software to access the hardware.

# Goals

- A 64 bit OS that people can use
- An OS that people WILL use
- An Os written completely in FASM assembly
- A stable OS that won't crash
- An OS that gives two options to the programmer low-level or high-level

# Running Environment

The Operating System will be organized in a manner as such:




Most software will run in ring 0. The only protection for the software will be provided through the kernel. Software can use the kernels software interrupts, use the driverlib functions, or communicate directly with the hardware. This provides a number of options while maintaining a maximum of control over the computers functions. There will also be a ring 3 option available so that malicious software has less access

when you are connected to the internet. The ring 3 software will only be able to use the software interrupts to communicate with the kernel and drivers, direct hardware control will not be allowed, nor will access to the driverlibs. There will be a driver interface for the ring 3 software to use the hardware. The main use of ring 3 will be for things like connecting for the internet, it is not the main mode of the OS. The purpose of the OS does not revolve around networking and using internet, just the ability to explore the hardware easily. Device drivers are not needed at all unless you use ring 3.

## Scheduling

Scheduling will be cooperative multitasking while the running environment is in it's main mode. While in the optional ring 3 mode, the kernel will feature preemptive multitasking. The reason that cooperative multitasking has been chosen is so that the OS does not have to worry about two processes accessing the disk at the same time and sending different commands to the same ports at the same time. This way all tasks can finish their critical regions and not have errors. This puts all of the when to schedule decision on the programmer, though it is not expected that an inexperienced programmer will be using this system.

## Memory

Paging will be used in the memory scheme as BareMetal is a 64 Bit Operating System. The only real memory protection will be through what the system provides with the exception of the ring 3 mode of the OS.

## System Calls

Here is a list of all system calls that will be implemented as software interrupts, all other functions will be in the driverlibs. There will be two classes of system calls main and ring 3.

MAIN:

Description:

There are fewer main system calls than there are ring 3 system calls. This is because just about all of the main functions can be implemented in the driverlibs.

void schedule();

The schedule(); function is for when a process has gone past its critical region and can give up the CPU. All it does is execute the highest priority process.

RING 3:

Description:

Most of the system calls are for ring 3 as the driverlibs are unavailable and direct hardware access is not allowed. They mostly relate to read and write.

int read(.....);

The read(.....) system call will work to read from files and devices, while in ring 3 BareMetal treats all devices as files as is in UNIX.

int write(.....);

Same as read() except for the fact that it writes to files and devices.