BareMetal Design Doc

Introduction

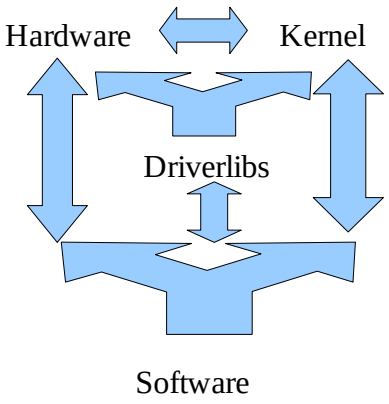
This is the design specification for the BareMetal Operating System. The first part of this design document will be a general overview of how the operating system works, the later part will be a detailed description of how specific things work like memory allocation and deallocation and such. BareMetal 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.

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:



All 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. The purpose of the OS does not revolve around networking and using internet, just the ability to explore the hardware easily.

Scheduling

Scheduling will be cooperative 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 and not know when to schedule.

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.

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.

Description:

There are very few system calls, this is because most all hardware access and such is done through the driverlibs. Just about all of these system calls relate to process management.

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.

```
execute(char *file);
```

The execute system call starts a new task. There are no flags options because even if protection is provided for files in the driverlibs the programmer can just not use them and write his own procedures to access a file that is owned by another user. All files are puplic.