Project Proposal

Title: A Multitasking Operating system

Student: Michael Rochester **Supervisor: Martin Berger**

Project Aims:

The purpose of this project is to produce a simple multitasking system in order to fully understand the working of OS's at the lowest level. I am interesting in this topic as an operating system sits at the heart of every computer. It defines the way that we as users interact with the system, and defines what kind of application we can run, this makes it one of the most important and central programs in computer science, and getting it right is not easy.

Primary Objectives:

- Create a bootloader to boot a kernel.
- Create a kernel that should be able to:
    - Load multiple 'programs' into memory.
    - Execute multiple 'programs' pseudo simultaneously using time sliced scheduling.
    - Allow multiple 'programs' asynchronous access to areas of the terminal screen.
    - Allow for the management of the execution state of each 'program'
    - Allow each 'program' access to a unique memory space.
    - Allow 'programms' to asynchronously send data to each other.
    - Handle all common cpu interrupts and resolve them correctly.
    - Handle keyboard input and allow programs to 'collect' keyboard inputs
- Create a small set of 'programs' with which to test the kernel.

Extended Objectives:

- Add some hardware drivers to load programs off disk and have some persistence.
- Move into pixel graphics mode.
- Use Multiple core's to have 'true' concurrency.
- Implement the entire c standard library.
- Port the 'Python' runtime to my OS (here I have chosen Python simply because there is a well documented 'todo' list for porting python to new OS's).

Relevance.

Computer science is the study computers, how they work, and what we can use them for. Operating Systems are the core of any computer. They define the what and the how of computer functionality. They sit at the very heart of all computer sciences.

Resources required:

- A computer for code development (already acquired)
- A i386-elf compiler for code compilation (already acquired)
- A i386 x86 emulator for code execution (already acquired)

Optional resources:

- A Physical i386 cpu and related hardware for code execution (already begun attempting to acquire). Although it would be nice to boot my kernel directly on the hardware, It is not necessary for the completion of the project.

Background Reading:

The C Programming Language (2nd edition) – Brian E. Kernighan & Dennis Ritchie.

This book provides a details description of the C programming language. As this is the language I have chosen to implement my OS in it is crucial I understand it thoroughly.

Modern Operating Systems (Jul 2013) – Andrew S. Tanenbaum.

This book describes key concepts and Algorithms behind operating systems, It will be a useful resource on understanding how current systems solve problems.

http://wiki.osdev.org/Main_Page (and all linked pages.).

This website provides detailed explanations of each component of an operating system from the perspective of someone implementing it. So far it has been an invaluable resource to understand how each component should behave.

Log:

Week 1:

Created basic design docs.
Rewrite existing code to be suitable for this project.

Week 2:

Detect Upper Memory.
Test Stack Jumping.
Simple page frame allocator.
Bound all interrupts to stubs.
Created simple kernel threading.
Created a 'printf' like utility.
Supervisor Meeting 1:

This week we discussed the state of the project, some goals to hit in the upcoming weeks and the proposal report due in next week.

Week 3:

Implemented Paging.
Moved to a higher half kernel.
Loaded my first 'user space' program.

Time table:

Follows is my current time table. My designated Project work time is in Blue, however time during the Purple 'Coding' blocks will also be used for project work where necessary.