Project "HalOS"

Created on 11-2-2001

Requirements

1. Introduction

1.1. Purpose

The purpose of this document is to define the requirements that must be met by the end of project “HalOS”. These requirements are to be followed to ensure a consistent design in the system, and to provide a way of testing the requirements when the project is finished. These requirements have attempted to be feature oriented as much as possible. There are a couple of sections (mainly 2.2) that are technical. This document is not intended to inhibit the creativeness of the engineer that will be implementing the system. This document is to ensure the functionality of the system.

2. Software

2.1. Boot Loader

2.1.1. HalOS shall load the kernel to a designated memory location from a floppy disk transfer execution to the loaded kernel

2.1.2. HalOS place the IA-32 in to protected mode before transferring control to the Kernel

2.1.3. HalOS shall leave the system in a “known” state prior to transferring control to the Kernel. See Section 2.2 for more on the “Know” state**.**

2.2. Know State

2.2.1. The boot loader shall leave the Kernel with three global descriptor tables creating a flat memory model, these include code, data, and stack

2.2.2. The boot loader shall have segment registers point to the appropriate global descriptor tables

2.2.3. The boot loader shall have the IA-32 in protected mode

2.2.4. The boot loader shall have disabled interrupts

2.3. Kernel

2.3.1. Applications

2.3.1.1. HalOS shall support an applications running in the same memory space as the kernel.

2.3.1.2. HalOS shall allow all kernel system calls to be callable by the application developer

2.3.1.3. HalOS shall allow only ONE application to run in the system, but this application can host several threads, which in turn can be applications within themselves.

2.3.2. User Input

2.3.2.1. HalOS shall provide the ability to retrieve input form a keyboard.

2.3.2.2. HalOS shall provide the application developer with an API to access the data retrieved from the keyboard

2.3.3. User Input

2.3.3.1. HalOS shall provide the ability to retrieve input form a keyboard.

2.3.3.2. HalOS shall provide the application developer with an API to access the data retrieved from the keyboard

2.3.4. User Output

2.3.4.1. HalOS shall provide the ability to display data on the standard monitor.

2.3.4.2. HalOS shall provide the application developer with an API to display information.

2.3.5. Tasks

2.3.5.1. HalOS shall provide support for "threads" that run within kernel space.

2.3.5.2. HalOS shall provide the application developer with an API to create and destroy tasks as needed.

2.3.6. Mutual Exclusion

2.3.6.1. HalOS shall provide the ability to use "Semaphores" and "Mutexes" between tasks.

2.3.6.2. Shall provide an API to create, access, and destroy "Semaphores and "Mutexes".

3. Hardware/Interface

3.1. Exceptions

3.1.1. HalOS shall dump the processors current state to the standard output device and halt the processor in the event that the processor fires an exceptions

3.2. Keyboard

3.2.1. HalOS shall provide a software "driver" to interact with the keyboard device.

3.3. RS-232

3.3.1. HalOS shall provide a software "driver to interact with the RS-232 hardware (specifically the PC16550 UART)

3.4. Video Output

3.4.1. HalOS should provide a software "driver" to interact with the video device (specifically the memory mapped video of a standard PC) in text mode only

3.5. Block Devices

3.5.1. HalOS should provide a software “drive” to interact with the floppy drive. It shall support at a minimum of low level sector read and writes.