**Tutorial 1**

Crear un sistema operativo puede ser una tarea muy difícil o complicada de entender. Sin embargo, a lo largo de estos tutoriales se trata de dejar todos los conceptos fácilmente explicados para un entendimiento muy a manera general de como es que un sistema operativo asienta las bases para que toda aplicación pueda correr sin ningún problema y disponga de los recursos necesarios para ejercer sus tareas.

Los sistemas operativos no son un solo programa que corre en si, si no un conjunto de programas acoplados que nos proveen un entorno amigable para poder utilizarlo y ejecutar otras aplicaciones encima de el. Para poder crear un sistema operativo es necesario saber programar, de lo contrario no estaremos capacitados para hacer dicha tarea. Lenguajes como Ensamblador o C/C++ son utilizados para alcanzar dicho logro.

Para empezar a desarrollar sistemas operativos, no hay nada mas básico que utilizar el lenguaje ensamblador. Este lenguaje se comunica de “tu a tu” con el procesador y ejecuta las instrucciones tal y como el procesador las conoce. Por ser un lenguaje de muy bajo nivel, es el principal código en un sistema operativo ya que nos inicia el ambiente para poder prepara la entrada del “kernel” o núcleo del sistema, del cual también puede estar elaborado en ensamblador o utilizar un lenguaje de alto nivel como puede ser C o C++. Para efectos de este tutorial, se estará usando el ensamblador conocido como “NASM” y el compilador de C/C++ de Microsoft, Visual C++ 2005. Se pueden utilizar versiones mas recientes hasta la versión 2010.

Como ya se ha mencionado, el código que inicia el sistema operativo se llama “bootloader”. Este pequeño código debe de ir en los primeros 512 bytes del primer sector de un disco, ya se un flipa drive o un disco duro. Este programa es ejecutado por la BIOS y corre directamente sobre el procesador. Para efectos de este tutorial se estará usando un disco virtual de flipa al cual podremos cargarle cuantas veces queramos las versiones que vayamos desarrollando del bootloader y el kernel. Así ya tendremos listo la parte de donde almacenaremos nuestro desarrollo y desde done lo ejecutaremos. Ahora bien, para efecto de prueba, bien podemos usar una computadora real o utilizar un emulador. Aquí es donde entra BOCHS, un emulador de x86 y también un debugger que nos puede mostrar los registros en vivo según los vayamos programando con nuestro sistema operativo.

El proceso que seguiremos para desarrollar este tutorial será el siguiente:

1. Setera nuestro entorno de desarrollo (NASM/Visual C++/BOCHS/)
2. Crear el bootloader (Compilar con NASM y copiar a nuestro virtual floppy)
3. Crear el kernel en formato ELF o PE y ejecutarlo con nuestro bootloader desde el floppy drive.