Multibootloader en un Sistema Operativo Didáctico.

Nicanor Casas[1], Graciela De Luca[1], Sergio Martín[1],
Andrea Otero[1], Guillermo Barrett [1], Matías Alvarez[1], Romina Leguizamón[1]

[1] Universidad Nacional de la Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas
Dirección: *Florencio Varela 1703* - Código Postal: 1754
San Justo, provincia de Buenos Aires - Argentina
Teléfono: *4480-8900/8835*

{ncasas, gdeluca, smartin}@unlam.edu.ar
andy.otero, legui.romina @gmail.com
guillexx_2000@hotmail.com
ma_ez_al@yahoo.com.ar

Abstract. The present work of research is focused on the accomplishment of a study of feasibility to develop a MULTIBOOTLOADER that it allows to recognize, together with other operating systems to the SODIUM. It show here the advances and also impeded found for the implementation of the same one, such as recognizing operating systems like Linux or Windows, the installation of the multibootloader, to allow to return to the previous state to the installation, to optimize the space in memory. We will specify the new functionalities that they were adding in the development of the project. Taking as base the development already implemented by pupils of the matter one adds functionalities which can be reusable in other SODIUM's applications. Also it puts on emphasis in the improvement of the performance for future modifications.

Keywords: System Startup Manager, Multibootloader, Master Boot Record, Disk Partitions, Operating Systems

Resumen. El presente trabajo de investigación se centra en la realización de un estudio de factibilidad para desarrollar un MULTIBOOTLOADER que permita reconocer, conjuntamente con otros sistemas operativos al SODIUM. Presentamos aquí los avances como así también dificultadas encontradas para la implementación del mismo, como reconocer sistemas operativos como Linux o Windows, la instalación del multibootloader, permitir volver al estado anterior a la instalación, optimizar el espacio en memoria. Especificaremos las nuevas funcionalidades que se agregaran en el desarrollo del proyecto. Tomando como base el desarrollo ya implementado por alumnos de la materia se agrega funcionalidades las cuales pueden ser reutilizables en otras aplicaciones de SODIUM. También se pone énfasis en la mejora de la performance para futuras modificaciones.

Palabras clave: Gestor de arranque, Multibootloader, Master Boot Record, Particiones de Disco, Sistemas Operativos

1 Introducción

SODIUM es un sistema operativo desarrollado con propósitos didácticos, la instalación del mismo está en sus fases finales de desarrollo, y para la implementación se necesita la convivencia con otros sistemas operativos como Windows y Linux.

Para los alumnos que necesiten o deseen ver su funcionamiento esto puede ser un problema ya que los multibootloaders disponibles no lo reconocerían en el momento del booteo. Si bien ya se había desarrollado un pequeño multibootloader que reconozca SODIUM, desarrollado con anterioridad, éste necesitaba modificaciones y mejoras en su funcionamiento.

Es por ello que como propuestas se fijó el objetivo de mejorarlos y agregarle funcionalidades como una instalación independiente del SODIUM, además encontrar la posibilidad de volver al estado anterior a su instalación, es decir, realizar un Backup del MBR.

Como principal desafío se encontró la dificultad de reconocer Linux, la cual se resolvió utilizando el backup realizado antes de la instalación. Además se desarrollo la ejecución del multibootloader en modo Live y debido a la naturaleza de SODIUM de mejora continua, se propuso desarrollar la mayor parte de lo implementado en lenguaje C, lo cual facilitaría las futuras modificaciones.

Además se implementaron estructuras y funciones para manejo de disco que pueden ser reutilizables en otras aplicaciones dentro de SODIUM.

2 Análisis

Antes del comienzo de la realización del diseño, se investigaron los temas relacionados con impacto directo sobre el diseño del MultiBootLoader, tales como, GRUB, GRUB2, MBR, Particiones, BIOS. La solución propuesta se basó en la aplicación desarrollada anteriormente, con lo cual también se tuvo que analizar el código existente a fin de que la funcionalidad agregada no impactara de forma negativa a lo ya realizado y poder reutilizar funciones ya desarrolladas

2.1 Master Boot Record

De todos los sectores de una unidad de disco, el primero de la primera cabeza del primer cilindro tiene una gran importancia ya que es el sitio al que se dirige la BIOS cuando busca si existe en el sistema un dispositivo cargable. Sus 512 bytes contienen tres bloques con información sobre la arquitectura física y lógica del disco: el código maestro de carga, la tabla de particiones y la firma.

Código maestro de carga: Si el disco es *"booteable"*, los primeros 446 bytes del MBR (sector de arranque), están ocupados por un pequeño trozo de código denominado *código maestro de carga MBC* ("Master Boot Code") o cargador inicial (bootstrap loader), que es cargado por la BIOS para comenzar el proceso de carga. El bootstrap loader lee la tabla maestra de particiones buscando una partición activa. En

caso de encontrarla, busca su sector inicial, carga su código en memoria, y le transfiere el control. Dicho código es ya capaz de cargar y ejecutar cualquier otro programa situado en cualquier partición del disco, que a su vez inicializará directamente el SO, o tal vez una utilidad conocida como gestor de arranque, que permite elegir entre distintas alternativas.

A continuación del MBC, se sitúa la *tabla maestra de particiones MPT* ("Master Partition Table"). Está constituida por cuatro trozos de 16 bytes (4 entradas) que contienen información sobre las particiones definidas en la unidad.

Para que un disco "maestro" (cuyo MBR pueda ser accedido por el proceso de carga de la BIOS) sea "booteable", se precisa que alguna de las entradas de la tabla contenga un 80h en el byte de estado, señalando cual es partición activa.

Los dos últimos bytes del sector de arranque (MBR) contienen dos caracteres (55h, AAh). que son denominados *firma del sector de carga* ("Boot record signature")

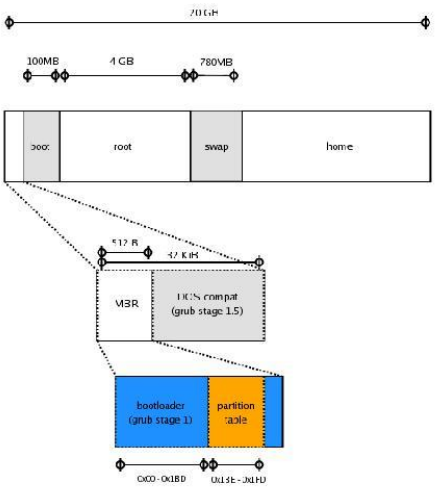

Fig. 1.

2.2 Particiones

La estructura lógica de segundo nivel que puede encontrarse en un disco físico es la partición primaria, esta es la forma en la que el sistema puede considerar dividida la unidad física en varias partes, que pueden incluso tener distinta estructura, donde cada una de estas particiones puede albergar un sistema de archivos distinto. Tales

como, un sistema FAT de DOS/Windows o un sistema ext2 Linux. Además de otras ventajas, las particiones ofrecen la posibilidad de montar distintos sistemas operativos, lo que a su vez, permite al equipo adoptar "personalidades" distintas según el SO que se cargue en cada momento.

En cada unidad se puede tener una partición primaria y una extendida, donde la partición primaria permite cuatro posibilidades del disco, por su parte la extendida consume otras cuatro posibilidades.

2.2.1 Estructuras:

La parte superior corresponde al MBR ("Master Boot Record"), a continuación siguen tres particiones primarias que son de estructura muy sencilla; solo tienen sector de carga ("Boot sector") y área de datos. El área de datos tiene composición análoga en todos los casos; comienza con las tablas FAT; sigue con la estructura del directorio raíz y a continuación los datos propiamente dichos.

En principio una partición extendida (extended Disk partition) no debería ser distinta de las demás particiones primarias de la unidad. Sin embargo, se le incluyeron algunas estructuras adicionales para poder subdividir su espacio en partes más pequeñas denominadas volúmenes lógicos VL. Lo mismo que la información de la *Partición Primaria de Disco* y de cualquier otra partición primaria que se hubiese definido en la unidad, la información básica sobre la *Partición Extendida de Disco* está en la entrada correspondiente de la MPT del sector de arranque. A su vez, el primer sector de la *Partición Extendida de Disco* es de funcionalidad y estructura semejantes a las del bloque de carga MBR, aunque en este caso se denomina EBR ("Extended Boot Record"). La imagen anterior muestra la organización interna de una partición extendida que contiene tres volúmenes lógicos. La tabla de particiones del EBR se denomina tabla de partición extendida xPT ("Extended Partition Table"), y tiene la misma disposición interna que la tabla MPT.

La primera entrada de la xPT señala a su propio sector de carga "Boot sector", la segunda señala al EBR del siguiente volumen lógico (si no existiera ningún otro, contiene ceros). Las 2 entradas restantes no se utilizan (contienen ceros). El segundo volumen lógico tiene una estructura similar al primero, la segunda entrada de su xPT señala a la EBR del siguiente y así sucesivamente. Puede verse que la estructura de los Volúmenes Lógicos es una cadena de enlaces que comienza en la MPT, y que cada volumen lógico contiene su propio EBR.

Cuando se declara una *Partición Extendida de Disco*, inicialmente el espacio está vacío, pero puede habilitarse construyendo en su interior una o varias particiones lógicas ("Logical partitions"), también denominadas volúmenes lógicos VL. (Fig.2) Posteriormente los sistemas DOS/Windows las referencian con una letra, de la C a la Z, como si fuesen discos distintos. Las letras A y B se reservan para designar los sistemas de fichero de los disquetes (que no admiten particiones). Durante el proceso de carga, el SO realiza una exploración ordenada de las unidades de disco y sus volúmenes lógicos. Los sistemas DOS/Windows asignan sucesivamente una letra a cada volumen encontrado. Por esta razón, en una Partición Extendida de Disco pueden declararse hasta 24 volúmenes lógicos si no se está usando una Partición

Primaria de Disco, o 23 en caso contrario. El factor limitador es la letra que se asigna a cada VL (empiezan por C y terminan con la Z). Por supuesto que salvo circunstancias excepcionales, es muy raro que se definan tal número de VLs en una partición extendida DOS.

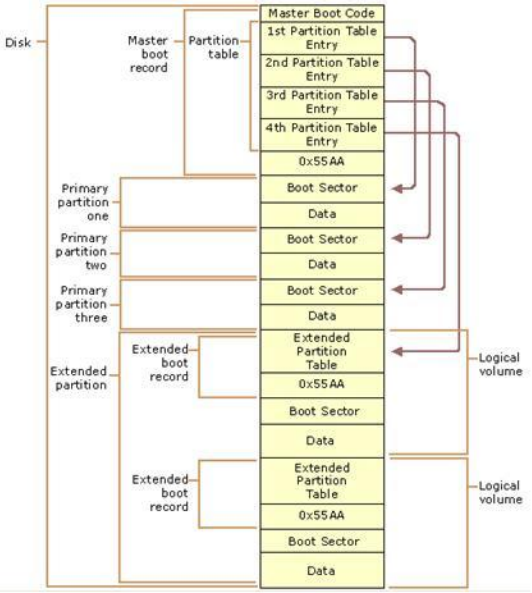

Fig. 2.

2.3 GRUB.

El ***GR****and* ***U****nified* ***B****ootloader* (**GRUB**) es un gestor de arranque múltiple que se usa comúnmente para iniciar dos o más sistemas operativos instalados.

Mientras los gestores de arranque convencionales tienen una tabla de bloques en el disco duro, GRUB es capaz de examinar el sistema de archivos, no siendo este el único, ya que existen otros como LILO y SYSLINUX.

2.4 GRUB 2

Está basado en un proyecto llamado PUPA cuya meta era desarrollar un GRUB más "limpio", seguro, robusto y potente así como más portable y con soporte para diferentes idiomas, sin embargo *Grub2* es un paquete nuevo que permite:

- Scripting, condicionales, bucles, variables y funciones.
- Interfaz gráfica.
- Extensibilidad mediante carga dinámica de módulos.
- Portabilidad a distintas arquitecturas.
- Internacionalización. Soporte para caracteres fuera del conjunto ASCII, mensajes
- localizados, etc.
- Mejor administración de memoria.
- Marco de trabajo modular, jerárquico y orientado a objetos para sistemas de archivo, archivos, dispositivos, unidades, terminales, comandos, tablas de partición y cargadores de SO.
- Instalación multiplataforma.
- Modo de rescate para casos en los cuales es imposible iniciar.
- Corregir errores de diseño de la versión anterior de GRUB, que no pueden resolverse debido a compatibilidad inversa, por ejemplo el numerado de las particiones

3 ALCANCES

Tomando como base el trabajo realizado anteriormente en el MBUM (Fig.3), se mejorará y agregara funcionalidades, tales como:

1. Instalación independiente con respecto a SODIUM y cualquier otro SO, en cualquier partición.
2. Búsqueda de etapa 2 en una partición primaria o lógica del tipo 0x12. Se asignará este id en el momento de la instalación del MBUM.
3. Generar un back up del MBR anterior a la instalación del MBUM. Este permitirá ejecutar el GRUB, Windows o SO instalados hasta ese momento, dependiendo de cómo se arrancaba antes de la instalación de MBUM.
4. Linux se iniciará a través del GRUB con el back up del MBR anterior. La misma va a ser temporal debido a la complejidad que tiene la carga de Linux, ya que es necesario leer el file system para encontrar la imagen de Linux y este no se encuentra en un lugar fijo.
5. La carga de Windows se permitirá sólo si la partición está marcada como Activa, debido a que el id de la partición no define si hay o no un SO. Cuando ejecute el Windows activo dará la opción de ejecutar otras versiones de Windows si están instaladas.
6. Ejecución en modo Live desde disquete, que permitirá la instalación del MBUM o cargar los sistemas operativos encontrados.
7. Una vez instalado MBUM permitirá la carga de los siguientes SO: SODIUM, Windows si esta marcado como activo, y el MBR anterior.
8. Crear una partición SODIUM nueva a partir del espacio nuevo no asignado, pasándole el ID y el tamaño para la creación.
9. Borrar una partición existente.
10. Crear un “fdisk” para SODIUM reutilizando el conocimiento y funcionalidad ya programada para 16-bit (Listar particiones y sus tipos, crear una partición a

partir de una partición de disco rígido hasta el siguiente final de partición, obtener los datos necesarios para montar/formatear/utilizar una partición de la lista).

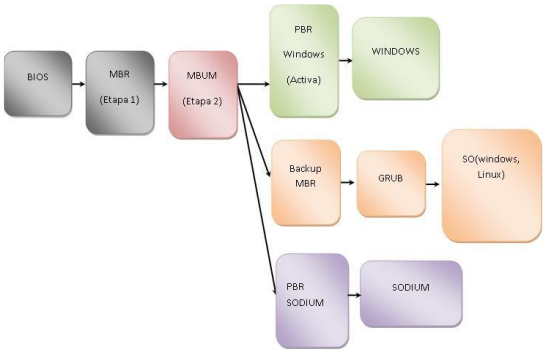

Fig. 3.

4 SUPOSICIONES Y RESTRICIONES

El principal inconveniente es la instalación del sistema operativo, en el momento en el cual se debería instalar el Multibootloader., dado que esta funcionalidad aún no está implementada en SODIUM por lo tanto solo se limitó a desarrollar la instalación del multibootloader independientemente a la del sistema operativo.
Como requisito para la instalación se tendrá que crear una partición primaria o lógica dedicada, en la que se instalará en la segunda etapa. Para la instalación del MBUM es necesario contar con las extensiones del BIOS, las cuales se utilizarán para leer y escribir en disco, para lo cual se agregó una validación para que no se pueda instalar el MBUM en caso de no contar con éstas.

5 DISEÑO

Durante el desarrollo del MBUM se encontraron dos inconvenientes:

a) No existe una forma estandarizada de saber que particiones son ejecutables, se puede obtener el tipo, pero no, si dentro de ésta se encuentra un SO al cual cederle la ejecución. Para dar solución a este punto el MBUM tomará como ejecutable las particiones primarias de tipo Windows que estén marcadas como activas y las particiones primarias o extendidas de tipo SODIUM.
b) Para realizar el booteo de Linux es necesario leer el sistema de archivos en el que se encuentra instalado, ya que no utiliza ningún mecanismo de carga en cadena como lo hace Windows. La solución momentánea encontrada a este problema es ofrecer la posibilidad de bootear cargando el MBR anterior y cediéndole la ejecución.

5.1 Administración de memoria

El BUM se ejecuta en el segmento 0x8000, el PBR carga el MBUM propiamente dicho a partir del offset 0 de ese segmento, de esta manera no ocupa lugar, dado que las rutinas del BIOS y la pila están situadas en el otro extremo del segmento: 0xFFFF, es la dirección inicial de la pila de registros de activación la cual crece hacia abajo, además se eliminaron las referencias explicitas a memoria ganando mayor flexibilidad y aprovechamiento de la misma.

5.2 Partición BUM

El siguiente gráfico **Fig.4** muestra como queda organizada la partición seleccionada luego de la instalación. Por simplicidad y debido a que la cantidad de archivos que la partición alberga es fija y la mayoría de estos son de tamaño fijo se opto por no utilizar un file system sino una organización arbitraria y colocar el BUM.bin al final para que pueda crecer todo lo que sea necesario.

PBR BUM	Sector 0
Backup MBR original	Sector 1
Sector reservado para el archivo de configuración	Sector 2
BUM.bin (rutinas y estructuras utilizadas por el BUM)	Sector 3 en adelante

Fig. 4.

5.3 Proceso de instalación

Cuando se ejecuta el BUM en modo LIVE este permite, entre otras funciones, la instalación del BUM en forma definitiva en el sistema.

El proceso de instalación.

- Al iniciar tanto en modo LIVE con desde disco el BUM recorre el disco y genera la tabla de particiones del sistema. El comando instalar recorre la tabla de particiones y las lista dando la opción al usuario de elegir en que partición desea instalar el BUM.

- Se genera un Backup del MBR original del sistema en la partición elegida por el usuario.
- Se lee el archivo BUM.bin (contiene funcionalidades y estructuras del BUM) del diskette y se copia en la partición a partir del 3 sector de la partición.
- Se copia el archivo PBR.bin del diskette al primer sector de la partición.
- Se lee el archivo BUMMBR.bin del diskette (este contiene el MBR del BUM) y se copia en el primer sector del disco. Antes de copiar el MBR se le agrega la tabla de particiones, la cual se obtiene del MBR original.
- Por último se cambia el tipo de partición a 0x12 (partición tipo BUM)

5.4 Funcionamiento del BUM

Una vez que el PBR carga en memoria el BUM y le sede la ejecución, este recorre el disco y crea en el HEAP la tabla de particiones del sistema y la tabla de particiones ejecutables. Una vez creadas estas 2 tablas busca en el archivo de configuración el sistema operativo que debe cargar por defecto, de no existir un archivo de configuración toma por defecto la primera partición ejecutable, espera el tiempo indicado el archivo de configuración o el que tiene por defecto y ejecuta la partición indicada.

5.5 Tabla de particiones del sistema

Esta tabla contiene una entrada por cada partición que posea el disco maestro. Básicamente se guarda la misma estructura que tiene la tabla de particiones tanto para las particiones primarias como para las lógicas. Los cuatro primeros lugares de esta tabla se reservan para particiones primarias y a partir de la quinta posición se colocan las particiones lógicas a fin de poder distinguirlas. Esta tabla se genera en el HEAP.

5.6 Tabla de particiones ejecutables

Para generar esta tabla se recorre la tabla de particiones del sistema y se guardan el HEAP aquella que este marcada como activa y este en una partición primaria o aquellas que sean de tipo SODIUM y estén en una partición primaria o lógica.

Además si detecta que existe una partición de tipo BUM, lo cual indica que el BUM está instalado, agrega una entrada que permite ejecutar la opción de booteo anterior que básicamente carga el BackUp del MBR que se resguardo durante la instalación. De esta manera se da al usuario la posibilidad de bootear como lo hacía antes de la instalación del BUM.

6 CONCLUSIONES

Una vez desarrollado el MBUM y viéndolo funcionar, observamos que hemos llegado a una forma sencilla de instalar un multibootloader capaz de reconocer, por defecto, SODIUM sin necesidad de realizar modificaciones al SO. Permite la carga

de varios Sistemas Operativos facilitando de esta manera la tarea de prueba de SODIUM en entornos reales mientras se sigue desarrollando, dejando a la PC completamente utilizable para realizar otras tareas.

El hecho de que se encuentre programado en C va a facilitar futuras modificaciones. Además durante el desarrollo se crearon varias estructuras y funciones para manejo de disco que pueden ser fácilmente aplicadas a SODIUM.

Nos resta en el futuro realizar un módulo para reconocer distintos file systems para hacer en forma automática la carga de otros Sistemas Operativos, tales como Unix o Linux que actualmente se realizan mediante el Backup del MBR, e incorporar una interfaz gráfica para que los alumnos puedan ver cada uno de los pasos y estructuras utilizadas en el Booteo de un disco con varios sistemas operativos disponibles.

7 BIBLIOGRAFÍA

1. Russinovich, Mark E., Solomon D.A. **Microsoft Windows Internals** (eds.) Microsoff Press, Fourth Edition, pp251-255, pp 615-654 (2005)
2. Brey B., ***Los Procesadores Intel***, Prentice Hall, Quinta edición, 2000
3. Townsend, Scott - **BIOS Boot Specification Editorial** (1996)
4. Lathner, Chris, John Murphy -**The Boot Sector** *Papers* **– (**2000)
5. Landis, Hale **BIOS Types, CHS Translation, LBA and Other Good Stuff (**1995)
6. Turley J. **Advanced 386 Programming Techniques**, Editorial Osborne McGraw-Hills, Primera Edición, 2004

Referencias WEB.

7. Details of GRUB on the PC (2005) http://www.pixelbeat.org/docs/disk/
8. SUSE Documentación de LINUX. Capítulo 29. Cargador de arranque / 29.3. Arranque con GRUB (Novell Inc.) http://www.mafuso.net/susemanual//index.html
9. Sitio Oficial de Grub http://www.gnu.org/software/grub/
10. Manual de Grub desde GNU http://www.gnu.org/software/grub/manual/html_node/
11. Alonso, Saúl. "La programación del GRUB" monografía, Sistemas Operativos UNLaM 2009.
12. Zator Systems: Tecnología de la información para el conocimiento, http://www.zator.com/Hardware/H8_1_2c.htm
13. Sáez, Sergio, Pérez,Vicente Lorente, J. C. - Estudio de un Sistema Operativo – Universidad Politécnica de Valencia - http://futura.disca.upv.es/~eso/