Problemáticas del arranque en Pendrive en un sistema operativo didáctico.

Hugo Ryckeboer, Nicanor Casas, Graciela De Luca, Martín Cortina,
Gerardo Puyo, Waldo Valiente
{hugor, ncasas, gdeluca, mcortina, wvaliente}@unlam.edu.ar
Pablo Pietropaolo, Pablo Sandler, Leandro Sposato, Sergio Tiraferri
{Pietrosoft,pablosandler}@gmail.com, {leandrosposato,sat_86}@hotmail.com

Universidad Nacional de la Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas
Dirección: *Florencio Varela 1703* - Código Postal: 1754 - Teléfono: *4480-8900/8835*

Abstract: The aim of this research project is to know about the involved details in the SODIUM operative system starting point from a USB flash support. On this investigation it is essential the structural analysis of the devices from a physical and a logical point of view. Base on this, we found differences among the technologies used for its construction and its internal functioning.
The other thread of the exploration is based on the different types of BIOS and their influence on the boot. There is no available BIOS standard in the industry and, despite the fact that a common behavior is expected for all of them, each supplier can emphasize differences in the functioning. These two axes are the conductors of the entire research, together with the particular features of the development surroundings (as the Linux tools), and the SODIUM.

Palabras clave. BIOS, USB, Boot, Logical Structure, Flash Memory, Master Boot Record, Partition, LBA, CHS.

Resumen. Este proyecto de investigación tiene como fin conocer los detalles involucrados en el arranque del sistema operativo SODIUM desde un soporte flash USB. En esta investigación se hace fundamental el análisis estructural, tanto físico como lógico de los dispositivos. A partir de esta base encontramos diferencias en las tecnologías utilizadas en su construcción y su funcionamiento interno.
El otro hilo de la exploración se basa en los diferentes tipos de BIOS y su influencia en el arranque. Al no existir un estándar de BIOS en la industria y a pesar de que podría esperarse un comportamiento común para todos, cada proveedor puede marcar diferencias en el funcionamiento, Son estos dos ejes los conductores de la investigación toda, acompañados por las características particulares del entorno de desarrollo (como las herramientas brindadas por el sistema Linux), y mismo el SODIUM.

Palabras clave. BIOS, USB, Boot, Estructura Lógica, Memorias Flash, Master Boot Record, Partición, LBA, CHS.

1. Introducción

Una parte importante del proceso consistió en la prueba de los parámetros que involucran a cada uno de los aspectos nombrados anteriormente, lo que arrojó una tabla de resultados y un informe de compatibilidad finales.

Al ser el SODIUM un desarrollo especialmente orientado a la enseñanza y al estudio de los sistemas operativos, resultó motivador para el grupo poder encarar las modificaciones necesarias para que pudiera ser instalado y ejecutado desde Pendrives u otros dispositivos USB. Esta nueva característica que habrá de adquirir el sistema operativo al finalizar el trabajo, permitirá una instalación más simple, económica y segura por parte de los alumnos que continúen con el estudio y el desarrollo del mismo.

Debido a que un host USB debe manejar el bus, la programación de éste se hace más compleja que programar el dispositivo USB. Pero, para algunas aplicaciones, la capacidad de almacenar datos en unidades genéricas hace que valga la pena la mayor complejidad.

Todas las unidades USB y otras unidades modernas soportan direccionamiento de bloque lógico (LBA). Con LBA, los bloques con capacidad de almacenamiento se numeran secuencialmente y comienzan en cero. Todos los bloques tienen el mismo tamaño, típicamente 512 bytes.

La dirección del bloque lógico se refiere a menudo como una dirección del sector porque el tamaño del bloque es igual a la capacidad de un sector en un disco duro. Para acceder al medio, el software especifica la dirección de bloque lógico para leer y escribir.

Para los discos duros, el controlador de la unidad traduce cada LBA a un cilindro, cabeza, y sector en el disco. En cambio, para las unidades flash, el controlador de la unidad traduce cada LBA a un bloque, página, y columna de la matriz de memoria.

La secuencia de direcciones de bloque lógico no tiene por qué corresponder a las ubicaciones físicas de los sectores en una unidad o la memoria en un chip. Todo lo que importa es que el controlador de medios sepa qué área de almacenamiento corresponde a cada dirección.

En los sistemas más antiguos, el software de acceso a los medios de almacenamiento usaban direccionamiento CHS, donde el software especifica un cilindro, la cabeza y número de sector para leer y escribir. Un dispositivo de almacenamiento puede soportar tanto direccionamiento CHS como LBA. [AXE96]

2. Elección de la Tecnología

Existen dos tecnologías de memoria flash en el uso popular: NOR y NAND y sobre las cuales el equipo tuvo que tomar una decisión. Si bien se deben incorporar las dos al SODIUM era imprescindible poder definir cuál de las dos sería la primera en ser incorporada.

NOR flash es ideal para almacenar el código del programa, donde la CPU requiere rápido acceso de lectura, pero rara vez se escribe en la memoria. NOR flash tiene

tiempos de lectura rápida pero tiempos lentos para borrar y escribir. Además, tienen una densidad baja y (para cantidades grandes de almacenamiento) puede requerir múltiples chips. Para acceder a las NOR flash, una CPU utiliza la mismos datos y líneas de dirección que utiliza para acceder a otros chips de memoria en paralelo. Dado que este es un sistema operativo el máximo tiempo está dado por la lectura de datos que se instalan en el disco duro o en memoria y generalmente no se utilizan grandes escrituras de datos por lo que se eligió esta como primer elemento a incorporar.

Los dispositivos de almacenamiento flash NAND, poseen tiempos de borrado y escritura mucho más rápidos. Flash NAND también tiene menor consumo de energía y es mucho más barato que las NOR, sin embargo esta quedo pospuesta debido a que es más compleja la programación de esta última

El código de corrección de errores (ECC) permite verificar los datos. El controlador genera y almacena un ECC cada vez que se graba un bloque; luego, durante la operación de lectura, se utilizar el código para verificar que los datos hayan sido leídos correctamente.

Los fabricantes de tarjetas de memoria flash pueden poner en práctica protocolos adicionales para ayudar a garantizar la fiabilidad de los datos. Por ejemplo, en condiciones de margen, el controlador en un SanDisk MultiMediaCard, luego de la escritura, lee los datos para verificar que la operación haya sido correcta. Si un bit es incorrecto, el controlador reemplaza el bit con uno de recambio. El controlador también puede sustituir un bloque incorrecto entero con otro bloque. Los protocolos de nivel superior también pueden apoyar a través de sumas de comprobación, o bits de corrección de errores enviados con los datos.

3. Estructura lógica

Se generó con el protocolo para dispositivos SCSI la interfase paralela siguiendo los estándares para dispositivos de almacenamiento que utilizan otras interfaces de hardware, incluyendo USB. [AXE96]

Las unidades flash y discos duros tienen hardware diferente, pero comparten muchas de las mismas estructuras lógicas para gestionar el almacenamiento de los datos, así como el Firmware para escribir o leer en el formato de los medios. Esto nos llevó a generar una estructura mediante la cual pudiésemos generar un mecanismo que nos permitiese pasar de una a otra con un mínimo de programación y fácil comprensión para los que deban seguir la secuencia.

Las estructuras lógicas, los métodos de direccionamiento y los sistemas de archivos descritos aquí tienen su origen en IBM PC, sus derivaciones y los sistemas operativos desarrollados por Microsoft para usarse en esas computadoras.

4. El sector de arranque USB

El sector de arranque especifica qué sectores están disponibles para el almacenamiento de archivos, directorios, y qué sectores contienen las tablas de asignación de archivos. También pueden contener el código del programa utilizado en el arranque del computador.

La FAT mantiene un registro de los clusters de datos que utilizan los archivos.

El documento que define los tres sistemas de archivos FAT se titula “Sistema de archivos FAT32. Especificación de Microsoft”

Es el sector cero, es el sector Master Boot Record (MBR), que contiene 3 ítems: un área para código ejecutable, una tabla de particiones, y una firma de booteo.

La firma del sector de booteo está en el ítem final del sector MBR. El Byte 510 (1FEh) debe tener el valor AAh y el byte 511 (1FFh) el valor 55h. [AXE96]

La tabla de particiones permite definir una o más particiones (volúmenes lógicos) en los medios de almacenamiento. Muchos dispositivos tienen un solo volumen. La tabla de particiones en el sector de arranque tiene espacio para cuatro entradas de 16 bytes que sirven para especificar a que partición pertenecen los sectores.

La tabla está entre los 446 bytes hasta los 509. Una entrada puede comenzar en el byte 446, 462, 478 o 494. Cada entrada de la partición tiene campos que definen la ubicación de comienzo de la partición a la hora de direccionar a través de los métodos CHS o LBA.

El campo de LBA en el byte 8 especifica el sector de inicio de la partición expresado como un desplazamiento desde el comienzo del medio físico (el sector de arranque) y siguiendo las especificaciones definidas tuvimos en cuenta, en los cálculos que los valores CHS se ignoran cuando se usa LBA.

Si una partición es booteable, el código ejecutable en el MBR puede utilizar direcciones CHS para localizar el código de arranque en la partición.

El campo “tipo de partición” en el byte 4 especifica un sistema de archivos, también brinda información a cerca del tipo de partición.

Los tipos de partición 0Ch y 0Eh deben ser compatibles con LBA para permitir el uso de la interrupción 13h del BIOS para acceder al medio. Si la partición es tipo 00h, la entrada no se utiliza y la partición no existe.

El elemento final de una entrada de la tabla de partición, 12 bytes, es el número total de sectores en la partición. La mayoría de los programa omiten este valor y utiliza en su lugar, un valor equivalente almacenado por el sistema de archivos.

Los dispositivos con múltiples particiones pueden usar particionamiento compartido, donde una de las entradas de la tabla de partición es para una partición extendida, cuyo primer sector contiene como estructura un registro de inicio extendido (EBR), con su propia tabla de partición.

La tabla de partición en el MBR contiene información acerca de la partición primaria del medio. La tabla de particiones en EBR puede almacenar a más de una entrada para particiones secundarias y una entrada para las particiones extendidas.

La partición extendida adicional, si existe, contiene su propio EBR y tabla de particiones. Los sectores EBR en particiones extendidas contienen una tabla de particiones y la firma, pero no el código ejecutable. Cualquier dispositivo con más de cuatro particiones debe usar particionamiento extendido.

5. El proceso de booteo de SODIUM

Durante el comienzo de la investigación, partimos desde el supuesto de que las implementaciones de los BIOS no eran un problema a tener en cuenta. Sin embargo, las informaciones recopiladas de Internet, libros y otras fuentes, nos indicaron que las implementaciones de cada uno de los fabricantes eran una variable de notable importancia en el objetivo planteado.

Al iniciar las modificaciones para el booteo desde Pendrive, SODIUM tenía ya implementado un único sistema de archivos: FAT-12, la primera de las versiones del grupo de sistemas de archivos FAT. Esto fue implementado de esta manera debido a la facilidad que brindaba la utilización de diskettes.

FAT-12 resulta muy apropiado para este tipo de dispositivos dado que los mismos tienen una capacidad de almacenamiento relativamente baja (1.44 MB para diskettes de 3 y ½ pulgadas) y FAT-12 crea bloques de tamaño adecuado en los diskettes.

Esto es notablemente diferente si se utilizan dispositivos de almacenamiento masivo tales como Pendrives o memorias flash. Los mismos poseen una capacidad mucho mayor (actualmente en el orden de los Gigabytes); por lo tanto FAT-12 ya no es el sistema de archivos mas apropiado (el principal problema está asociado con la baja cantidad de información que es posible almacenar en un disco en FAT-12).

La solución al tamaño de la unidad consiste en particionar el dispositivo creando un sistema de archivos compatible con las especificaciones actuales de SODIUM, no solamente en tamaño, sino con el resto de los parámetros necesarios. Al mismo tiempo dejar libre el espacio no utilizado para poder ser aprovechado por otro sistema de archivos, incluso por SODIUM en el futuro.

Una parte de nuestra investigación se basó en el estudio del booteo del SODIUM, buscando conocer aquellos archivos involucrados en el proceso, y susceptibles de ser modificados.

Al comenzar el booteo, el BIOS comienza leyendo el sector de arranque del dispositivo, o lo que es lo mismo, el archivo bootSector.bin, copiado en dicho sector durante la instalación. En este archivo se encuentra el código de precarga del SODIUM.

Para el correcto funcionamiento, se establecen algunas variables, como por ejemplo, la BPB (BIOS parameter block), que especifica los parámetros físicos del hardware. Además, se definen los parámetros básicos para el inicio del SODIUM, como la dirección del stack pointer. Su tarea consiste en buscar en la FAT del dispositivo el

archivo loader.sys, el cargador propiamente dicho. Una vez encontrado, lo carga en memoria y le cede la ejecución.

Es, ahora sí, loader.sys es el encargado de traer a memoria todos los archivos que necesita el SODIUM para su funcionamiento. Para saber cuáles son esos archivos, lee primero el archivo listado.bin, que contiene los datos de cada uno de ellos.

Uno de los archivos cargados, el sodium.sys, corresponde al kernel, y es en este momento en que se le cede ejecución. A pesar de que el booteo no culmina aquí, el alcance de nuestra investigación no contempla los pasos siguientes.

Teniendo en cuenta el proceso de booteo, el método desarrollado consiste en modificar los distintos parámetros de bootSector.bin (de la BPB, más precisamente), de forma tal de adecuarlos a los dispositivos USB.

Al no contar con un mecanismo de depuración, gran parte de los cambios fueron realizados siguiendo un proceso de "prueba y error", en el que cada parámetro es probado por separado, para evaluar su impacto en el booteo.

dbNroDispositivo, corresponde al número de dispositivo físico donde se instalará bootSector.bin. El dispositivo 0x00 corresponde al Floppy Disk y el 0x80.

Cuando se prepara una unidad para ser booteable (junto con su respectivo sistema de archivos), es necesario: Crear la partición y formatear la unidad con un sistema de archivos apropiado para la instalación del sistema operativo y copiar los archivos en un cierto orden, realizando algunas marcas en los primeros bytes del sector de booteo para que el hardware/BIOS reconozca la unidad como booteable y transfiera el control al cargador del sistema operativo.

SODIUM es un sistema operativo construido por alumnos de la UNLaM, sin bibliotecas u otros elementos externos. Para codificarlo, debuggearlo, compilarlo e instalarlo se utiliza el sistema operativo Linux como entorno de trabajo. Esto significa que, para preparar el sistema de archivos en el dispositivo USB e instalar el SODIUM, debimos utilizar comandos que se encuentren disponibles en Linux, sin recurrir a herramientas externas al entorno.

Lo que se busca con los parámetros, es modificar la geometría de la unidad USB para que cuente con dos cabezas y dieciocho sectores por pista, como los diskettes. De esta forma, se espera que los cambios a las estructuras lógicas sean mínimos. Al mismo tiempo, se establece la cantidad de cilindros en mil veinticuatro, para otorgarle al dispositivo la máxima capacidad de almacenamiento posible.

dd es un comando de Unix que se encuentra disponible desde la década del 70 y que ha sido adaptado e implementado a muchos otros sistemas operativos (entre ellos Linux), convirtiéndose en una herramienta indispensable para hacer copias de datos en forma binaria. Su función es la de copiar y convertir datos a bajo nivel, respetando el orden y las estructuras en la que se encuentran los datos. Es por eso que es usado para realizar copias desde / hacia los sectores de booteo.

dd utiliza una sintaxis diferente a la mayoría de los otros comandos de Unix / Linux. Esta sintaxis es de la forma keyword=value u opción=valor y debido a la popularidad que adquirió el comando esto no fue modificado en sus versiones posteriores.

Esta forma de escribir los parámetros se utiliza en el dd inclusive para indicar el nombre de los archivos de entrada o salida (input file / output file) mientras en otros comandos no se suele indicar el nombre del archivo con otra marca más que el nombre mismo. dd [lista con forma opción = valor]

Al igual que con sfdisk, en este trabajo describimos únicamente los parámetros utilizados. Para mayor información se puede consultar el manual de Linux / Unix (man) que provee información detallada sobre el comando.

El resto de los archivos se copia con el conocido comando cp.

Ejecutar estos dos pasos antes nombrados para preparar cada unidad, sería un método tedioso y propenso a errores si se ejecutara de forma manual. Por tal razón, se decidió crear un script semiautomático, que actúa como guía para el usuario a lo largo del proceso de instalación. Para hacerlo aún más simple, y con la idea de lograr la máxima integración con la estructura del SODIUM ya existente, manteniendo al mismo tiempo la compatibilidad, el script no será invocado de forma directa por el usuario, sino indirectamente mediante el parámetro “install_usb” pasado al Makefile principal.

6. Pruebas

Si bien ya habíamos realizado algunas pruebas durante la etapa de diseño, como parte de la implementación fue necesario hacer y formalizar la mayor cantidad de pruebas posibles con un doble objetivo: verificar la compatibilidad del arranque del SODIUM vía USB y elaborar un informe detallado indicando el hardware utilizado, detallando los casos en los que el arranque fue exitoso y los casos en los que no.

Uno de los principios básicos del testing dice que cuando se toman muestras, estas deben ser lo mas representativas posibles de la población. Entonces, de forma ideal, deberíamos probar el arranque del SODIUM en "uno de cada uno" de los modelos de equipos existentes que tengan capacidad de hacerlo desde un Pendrive.

Esto resultaría imposible en la práctica. Desde la aparición de la IBM PC (es decir mucho antes de la tecnología USB) hasta nuestros días, salieron al mercado innumerables modelos de motherboards y equipos informáticos.

Sin embargo, refiriéndonos al tema de este trabajo, sabíamos que los posibles problemas de carga de un sistema operativo por medio de USB estaban asociados a algunos BIOS en particular, especialmente aquellos que poseen las máquinas con BIOS de marca propietaria.

Entonces, haciendo una primera división, identificamos los más comunes y buscamos equipos que tuvieran BIOS de cada uno de esos fabricantes.

En oposición a esto, buscamos BIOS propietarias en los equipos de marca que eran candidatas a presentar problemas, por lo tanto también realizamos las pruebas en las PC de marcas más reconocidas: Dell, Compaq, HP, IBM / Lenovo, Commodore.

7. Tipos de BIOS

Acorde a la documentación consultada y las pruebas realizadas, se puede hacer una clasificación de dos tipos diferentes de BIOS observados.

IBM/Microsoft: Esta implementación soporta dispositivos de capacidad mayor a 8 GB, INT 13H y las funciones de lectura/escritura de AH=4xH y EDPT para los dispositivos 80h and 81h. INT 13H AH=08H devuelve L-CHS e INT 13H AH=41H y AH=48H se utilizan para la P-CHS.

Estos llamados utilizan direccionamiento LBA y soportan dispositivos con hasta 2^64 sectores. A pesar de utilizar LBA, no requieren que la interfaz de dispositivo soporte LBA.

Los llamados de la Int 13h, AH=4xH no están siendo implementados en muchos sistemas actuales. Además, no está estandarizado el comportamiento que una BIOS debería tener al momento de entregar una respuesta a un llamado Int 13H AH=08H, con un dispositivo de capacidad mayor a 8 GB.

El soporte de LBA en la interfaz del dispositivo puede ser automático o presentarse como una opción definible por el usuario en el programa de configuración del BIOS. [LAN95]

Phoenix: Al igual que el tipo IBM/Microsoft, soporta dispositivos de capacidad mayor a 8 GB, INT 13H y las funciones de lectura/escritura de AH=4xH y EDPT para los dispositivos 80h and 81h. INT 13H AH=08H devuelve L-CHS e INT 13H AH=41H y AH=48H se utilizan para la P-CHS. INT 13H AH=48H devuelve la dirección de FDPT (tabla de extensión creada por Phoenix).

Como el tipo anterior, los llamados utilizan direccionamiento LBA y soportan dispositivos con hasta 2^64 sectores, sin embargo, estos llamados no requieren que la interfaz de dispositivo soporte LBA.

El llamado INT 13H AH=48H retorna información adicional, como soporte de DMA, LBA, etc. Según la compañía, los BIOS tipo Phoenix son una extensión o mejora de los BIOS originales de IBM/Microsoft, por lo tanto, las implementaciones deben soportar el conjunto completo de llamados a INT 13H AH=4xH.

Como en su predecesor, los llamados de la Int. 13h, AH=4xH no están siendo implementados y no está estandarizado el comportamiento que el BIOS debería tener al momento de entregar una respuesta a un llamado INT13H AH=08H, con un dispositivo de capacidad mayor a 8 GB. [LAN95]

Sería normal pensar que existe una única conversión de L-CHS a P-CHS o una única conversión de CHS a LBA, pero no es así, esto se debe a que no existe ninguna documentación que estandarice dichos algoritmos.

Como consecuencia, no se puede confiar en que todos los BIOS y todos los sistemas operativos realicen estas conversiones de la misma manera.

El gran problema que sale a la luz, es que no existe ningún requerimiento sobre cuál implementación del algoritmo de conversión hay que utilizar, y es por esta razón que

podemos distinguir entre los 2 tipos de implementación más populares: la implementación Phoenix y la No-Phoenix.

8. Conclusión

Desde el comienzo se consideró a este proyecto como fundamental en el desarrollo del SODIUM. El aporte facilitaría la instalación y el desarrollo por parte de los futuros alumnos de la cátedra de Sistemas Operativos, y maximizando el esfuerzo en las tareas de programación de nuevas funcionalidades. Alentados por el desafío, comenzamos nuestra tarea.

A partir de nuestro proceso de investigación, hemos logrado conocer el funcionamiento interno, físico y lógico del medio de almacenamiento masivo denominado “Memoria Flash”.

Esta investigación estuvo asentada en distintos documentos de expertos en la materia, que sirvieron como fundamento teórico para nuestro avance.

El material bibliográfico sobre los distintos BIOS presentes en las máquinas actuales, a pesar de ser vasto e interesante, no fue suficiente. Este fue uno de nuestros principales obstáculos durante la investigación, ya que en el inicio del sistema, los únicos servicios disponibles son los prestados por el BIOS, pero éstos, al tratarse de sistemas propietarios y cerrados, encontrar información sobre los mismos se tornó extremadamente complicado. Por este motivo, nuestro método de prueba fue un proceso básico de “prueba y error”, buscando detectar las diferencias de bajo nivel que presenta cada uno de los BIOS durante el arranque.

Una de las diferencias encontradas fue la forma en que cada uno de los BIOS detecta a los dispositivos USB. Los IBM los interpreta como una suerte de HD removible. Al contrario, Phoenix los detecta como un Pendrive USB, es decir, se corresponde en mayor medida con la realidad, por lo que no es de extrañar que sea en estas máquinas en las que se presentaron la mayor cantidad de pruebas exitosas.

Al continuar con las investigaciones, se pudo establecer que estas diferencias en la detección de los dispositivos, tienen su impacto en la conversión entre el direccionamiento LBA y CHS. Como consecuencia, esto influye negativamente en el proceso de arranque.

Teniendo en cuenta las pruebas exitosas y las incompatibilidades encontradas, desarrollamos un método que permite compilar el SODIUM, preparar el Pendrive y hacer la copia de los archivos necesarios, para que el sistema operativo arranque desde el dispositivo USB.

9. Bibliografía

[AZZ04] Azzarito, Doug, Fred Bhesania, Jim Blackson, Mark Bohm, Robert Chang, Jason Chien, David Cho y otros - **Universal Serial Bus Mass Storage Specification For Bootability –** USB Implementers Forum - 2004

[TOW96] Townsend, Scott - **BIOS Boot Specification Editorial** - 1996

[AXE96] Axelson, Janet - **USB Mass Storage Designing and Programming Devices and Embedded Hosts -** Lakeview Research LLC - 2006

[LAN95] Landis, Hale **BIOS Types, CHS Translation, LBA and Other Good Stuff** - 1995

[SEL07] Séller, Dave, Dave Pierson - **Installing Red Hat Linux from a USB flashdrive –** IBM - 2007

[LAT00] Lathner, Chris, John Murphy -**The Boot Sector -** 2000

10. Papers

[001] Tecnick S.R.L - **Post beep codes** - Sede Legale- Via Della Pace 11, 09044,Quartucciu (CA), ITALY http://www.technick.net/public/code/cp_dpage.php?aiocp_dp=guide_beep_codes

[002] Paul D. Cook, Mitch Stone - Macintouch Reader Report - PC History http://www.macintouch.com/pchistory.html

[003] Oberon Community Platform - **Boot Manager -** ETH Oberon web site of the Native Systems Group located at the Computer Systems Institute, Department of Computer Science, ETH (Swiss Federal Institute of Technology). - http://www.ocp.inf.ethz.ch/wiki/Documentation/BootManager

[004] Sergio Sáez, Juan Carlos Pérez,Vicente Lorente - Estudio de un Sistema Operativo - Universidad Politécnica de Valencia - http://futura.disca.upv.es/~eso/

[005] OSDev.org - Operative System Developers - http://wiki.osdev.org/Main_Page

[006] Pen Drive Linux - **Boot and run Linux from a USB flash memory stick -** http://www.pendrivelinux.com/

[007] Daniel Faulkner **- Hello World Boot Loader -** http://www.osdever.net/tutorials/view/hello-world-boot-loader

[008] Dave Heller, Dave Pierson - **Installing Red Hat Linux from a USB flashdrive** - IBM Linux Technology Center - IBM Network Transformation Center – 2007 - http://w3-03.ibm.com/support/techdocs/atsmastr.nsf/Web/Techdocs