



Proyecto Integrador

Bootloader, MBR y Grub

16 de julio de 2015

Laboratorio de Redes y Sistemas Operativos
Departamento de Ciencia y Tecnología

Grupo
Daniel Wrytowski Juan Acosta Ríos Leandro Di Lorenzo

Universidad Nacional de Quilmes

Roque Sáenz Peña 352, Bernal

Buenos Aires, Argentina (B1876BXD)

Tel. (+54 11) 4365 7100 | Fax (+54 11) 4365 7101

<http://www.unq.edu.ar/> | info@unq.edu.ar

1. Booteo de una computadora

Cuando se enciende la computadora, el motherboard recibe energía y se prende. Al encenderse, el mother inicializa su firmware y trata de hacer andar el CPU. Si todo anduvo bien el CPU comienza a andar. En un sistema multi-procesador o multi-core se selecciona un CPU (al azar) como procesador de arranque (bootstrap processor, o BSP) que ejecuta el BIOS (Basic Input Output System) y el código de inicialización del kernel del sistema operativo.

El BIOS es el primer programa que se ejecuta en la computadora, su propósito fundamental es iniciar y probar el hardware del sistema y cargar un gestor de arranque o un sistema operativo desde algún dispositivo de almacenamiento.

La mayor parte de los registros del CPU tienen valores definidos en el arranque, incluyendo el EIP (puntero de instrucciones) que guarda la dirección de memoria de la instrucción que se está ejecutando por la CPU. La CPU luego comienza a ejecutar código del BIOS, que inicializa algunos de los componentes de hardware de la máquina. Hecho esto, la BIOS lanza el POST (Power on self test) que chequea varios componentes de la computadora.

POST es un proceso de verificación e inicialización de los componentes de entrada y salida en un sistema de cómputo que se encarga de configurar y diagnosticar el estado del hardware.

El POST involucra una serie de tests e inicializaciones, incluyendo descubrimiento de recursos -interrupciones, rangos de memoria, puertos de entrada/salida para los dispositivos PCI. Los BIOS modernos que siguen las especificaciones ACPI (Advanced Configuration and Power Interface) construyen unas tablas de datos que describen los dispositivos en la computadora; estas tablas son usadas luego por el kernel.

Después del POST, el BIOS intenta bootear un sistema operativo, que debe estar en alguna parte: discos rígidos, CD/DVDs, USBs, u otros dispositivos de almacenamiento. El orden en el que el BIOS busca el dispositivo de booteo es configurable por el usuario. Si no hay ningún dispositivo de booteo el BIOS termina con un mensaje de error del estilo «No hay disco de sistema o error de disco»; esto sucede por ejemplo cuando el disco rígido está roto y no hay otro dispositivo de booteo. Si en cambio el BIOS encuentra un dispositivo de almacenamiento en funcionamiento, entonces permite continuar con el procedimiento de booteo.

Ahora el BIOS lee el sector de los primeros 512 bytes del disco rígido. Este sector se llama MBR (Master Boot Record) y normalmente contiene 2 componentes principales: un pequeño programa de inicio específico de un sistema operativo en el comienzo, seguido de la tabla de particiones del disco. El BIOS a esta altura ya no toma parte de este proceso, sino que simplemente carga los contenidos del MBR en memoria y apunta el EIP a la primera posición donde alojó esos contenidos, para que se comience a ejecutar el programa que se debería haber traído del MBR.

2. Cómo trabaja un bootloader

El código específico en el MBR puede ser un MBR loader de Windows, de Linux (LILO, GRUB) o incluso un virus. Por otro lado, la tabla de particiones está estandarizada, es un área de 64 bytes con 4 entradas de 16 bytes describiendo como el disco fue dividido (de manera de poder tener varios sistemas operativos o volúmenes separados en el mismo disco).

Tradicionalmente el código del MBR de Microsoft busca en la tabla de particiones la partición marcada «activa», carga el sector de booteo de esa partición y ejecuta el código. El sector de booteo es el primer sector de una partición. Si algo sale mal al intentar leer la tabla de particiones generalmente se obtiene el mensaje «Tabla de particiones inválida» o «Falta el sistema operativo». El mensaje específico depende del «sabor» de MBR que haya en el disco.

Con el tiempo el booteo se volvió más sofisticado y flexible. Los bootloaders de Linux como Lilo y GRUB pueden manejar una gran variedad de sistemas operativos, filesystems y configuraciones. El código MBR no necesariamente sigue el esquema de «bootear la partición activa» como el descrito antes, sino que el proceso por el que pasa se parece mas al siguiente:

1. El MBR contiene en sí mismo la primera etapa del bootloader. Grub llama a esto «Etapa 1» (Stage 1)
2. Dado que el espacio en el MBR es muy chico, el código del MBR hace solo lo suficiente para cargar otro sector del disco que contiene código de booteo adicional. Este sector puede ser el sector de booteo de una partición, pero también podría ser un sector del disco «hard-codeado» en el código del MBR cuando fue instalado.
3. El código MBR junto con el código cargado en el paso anterior a continuación lee un archivo que contiene la «segunda etapa» (Stage 2 en Grub) del bootloader. El código de la segunda etapa luego lee un archivo de configuración (grub.conf en el caso de Grub) y luego presenta las opciones de booteo al usuario, o simplemente procede con la carga del sistema operativo si no se configuró mas de una opción de booteo.
4. A esta altura el código del bootloader debe cargar un kernel. Por lo tanto el bootloader necesita tener suficiente información sobre el filesystem para poder obtener el archivo del kernel de la partición de booteo. En Linux este archivo generalmente tiene un nombre parecido a «vmlinuz-x.x.x-xx». Entonces este archivo se carga en memoria y el EIP salta a la primera instrucción de booteo del kernel.

3. Introducción a GRUB

Como se explicó antes, el bootloader es el primer programa que ejecuta la computadora luego del BIOS. El bootloader es el responsable de cargar el kernel de un sistema operativo y transferirle el control de la computadora (como puede ser Linux, GNU Mach, Windows, etc.).

El kernel, en cambio inicializa el resto del sistema operativo (por ejemplo, el sistema GNU).

GRUB es un boot loader muy poderoso que puede cargar una gran variedad de sistemas operativos libres y propietarios (estos últimos usando una técnica llamada chain-loading en la que se referencia al sector de booteo donde se encuentra el bootloader del SO a bootear, de manera que este tome el control). GRUB sabe trabajar con varios sistemas de archivos y formatos de kernels, de manera que no es necesario que se guarde en el boot sector la información de la posición física del kernel en el disco, sino que se puede cargar un kernel solo especificando el nombre del archivo, disco y partición en el que está ubicado. Al bootear con GRUB uno puede elegir entre una interfaz por línea de comandos o una interfaz gráfica de menús.

Al usar la interfaz por línea de comandos uno puede tipear las especificaciones disco y del archivo del kernel manualmente. En la interfaz gráfica en cambio, uno solo selecciona el sistema operativo usando las flechas del teclado. El menú que se presenta en esta última interfaz se basa en un archivo de configuración preparado de antemano al configurar el boot loader.

GRUB tiene dos métodos de booteo diferentes. Uno de los 2 es cargar un sistema operativo directamente, el otro es hacer chain-loading (delegar el booteo a otro boot-loader que luego cargará otro sistema operativo directamente).

En general, la primera opción es la mas conveniente, porque no se necesita instalar o mantener otros bootloaders, y GRUB es lo suficientemente flexible para cargar un SO desde una partición o disco arbitrarios. Pero hay casos en que se necesita la segunda opción ya que GRUB no soporta nativamente todos los sistemas operativos existentes. A su vez, GRUB cuenta con la opción de loopback booting, que permite bootear desde una imagen iso en un CD/DVD o HDD. Sin embargo, con esta opción el mismo SO deberá ser capaz de encontrar su root para funcionar.

4. Booteo manual de un Linux con GRUB

Mas allá de que generalmente uno no se detiene a especificar opciones de booteo cada vez que enciende la computadora, es posible, y a veces muy útil, saber bootear desde GRUB en forma manual, desde la línea de comandos. GRUB tiene una infinidad de opciones para configurar en el booteo, pero aca solo vamos a mostrar y explicar brevemente como bootear un linux desde Grub. Es importante aclarar que para bootear desde la línea de comandos a veces es necesario conocer no solo los comandos y opciones que provee GRUB sino también la disposición de los filesystems

y discos en la computadora, ya que para bootear manualmente hay que especificar como mínimo cual será el disco y el root filesystem del que queremos bootear, así como el archivo del kernel. Es decir, hay que entender un poquito lo que se está haciendo y no solo copiar y pegar lo que muestra el manual. Aún así, hay algunos comandos que, entendiendo algunas ideas de GNU/Linux, pueden simplificararnos la vida si no tenemos mucha información sobre la computadora en la que queremos bootear.

4.1. Pasos para bootear linux manualmente.

1. Setear el dispositivo de root al disco en el que está alojado el archivo del kernel del sistema operativo.

```
grub> search --set=root --file /vmlinuz
```

2. Cargar el kernel usando el comando linux.

```
grub> linux /vmlinuz root=/dev/sda1
```

Si se necesitan especificar parámetros del kernel, solo hay que agregarlos al final de la línea. Por ejemplo se puede setear el acpi en off haciendo

```
grub> linux /vmlinuz root=/dev/sda1 acpi=off
```

Con Linux, GRUB usa un protocolo de 32 bits. Algunos servicios de la BIOS como el APM o EDD no están disponibles con este protocolo. En este caso se necesitaría usar el comando linux16

```
grub> linux16 /vmlinuz root=/dev/sda1 acpi=off
```

3. Si se usa un initrd (un ramdisk) entonces hay que ejecutar el comando ramdisk seguido de la ubicación del archivo

```
grub> initrd /initrd
```

4. Finalmente se ejecuta el commando de booteo (boot)

```
grub> boot
```

5. Configuración de GRUB para bootear desde MBR o partción de disco

El programa de edición de GRUB que vamos a mostrar es Grub Customizer. Provee una interfaz gráfica que permite modificar las características del GRUB. Entre sus funciones encontramos:

1. Elegir los kernels booteables
2. Elegir el kernel seleccionado por defecto
3. Decidir sobre el booteo automático y el tiempo de espera hasta que suceda
4. Elegir tipografía y colores de los textos
5. Elegir imagen de fondo y resolución general
6. Poder cargar configuraciones visuales bajadas de internet

Para instalarlo es necesario escribir en la consola los siguientes comandos:

```
$ sudo add-apt-repository ppa:danielrichter2007/grub-customizer
$ sudo apt-get update
$ sudo apt-get install grub-customizer
```

Y uno puede iniciarlo escribiendo en la consola

```
$ grub-customizer
```

Al hacerlo, lo primero que nos aparece es la pestaña que contiene la lista de kernels disponibles para bootear:

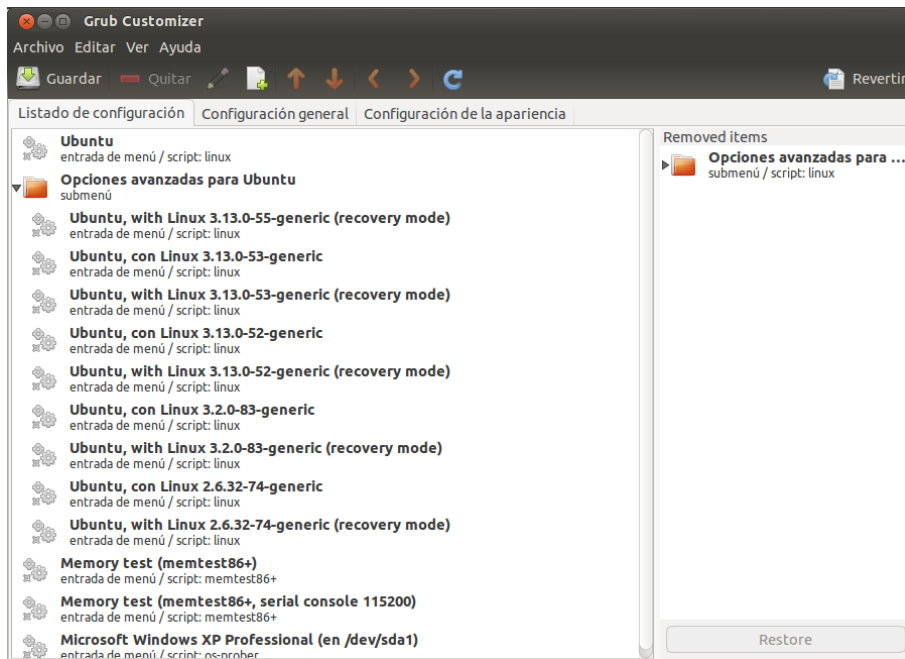
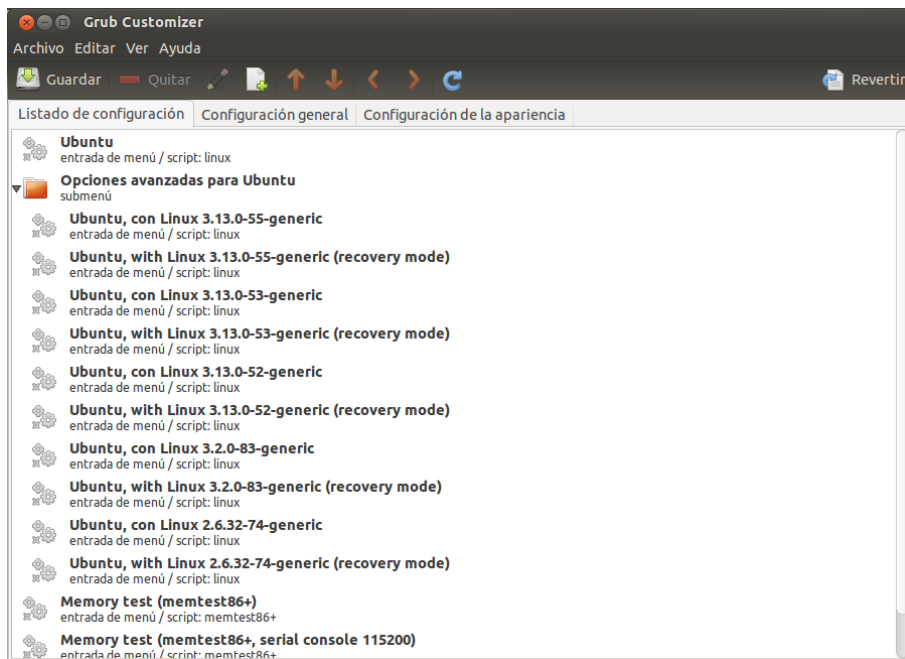
Si se selecciona alguno, uno puede quitarlo de la lista apretando el botón Quitar. Al hacerlo, nos aparece una segunda lista con los kernels removidos (pero no eliminados):

La segunda pestaña es la de la configuración general. En ella podemos modificar el kernel seleccionado por defecto, si queremos o no booteo automático y, en caso afirmativo, cuanto tiempo de espera hasta que se realice

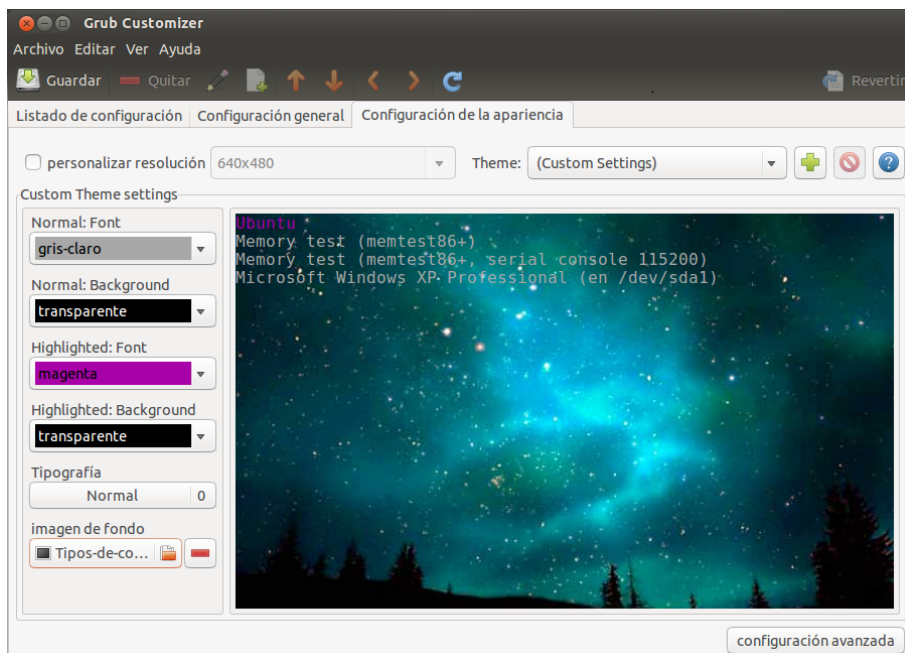
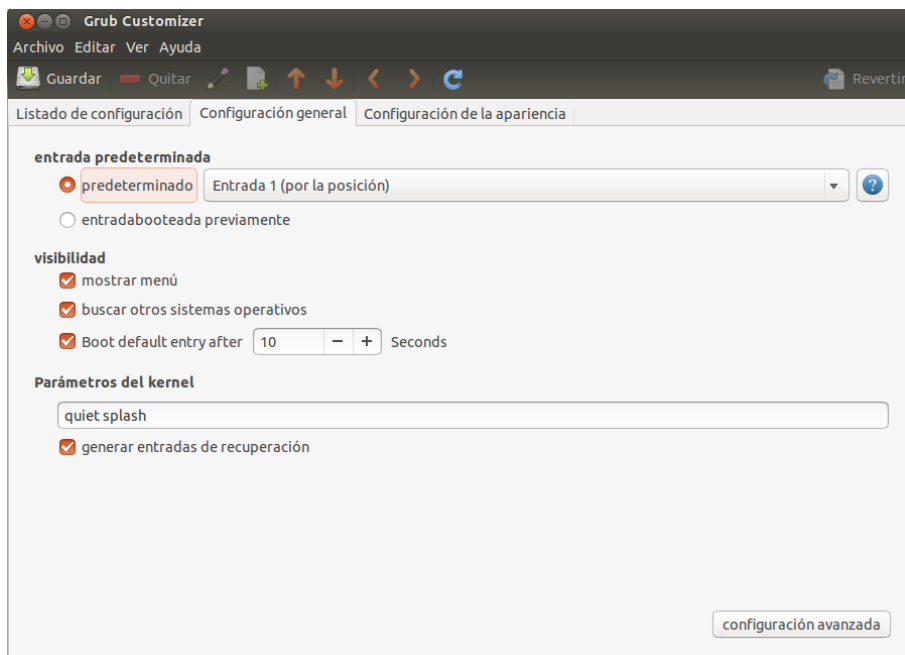
Por último, está la pestaña de configuración de apariencia. Podemos modificar en ella la resolución, la imagen de fondo, la tipografía y los colores de cada uno de los textos:

Para concretar todos estos cambios uno tiene primero que guardar el GRUB generado y luego modificar el MBR para que haga referencia a este nuevo GRUB. Esto se hace oprimiendo primero el botón Guardar y luego yendo a Archivo -¿ Instalar en el MBR:

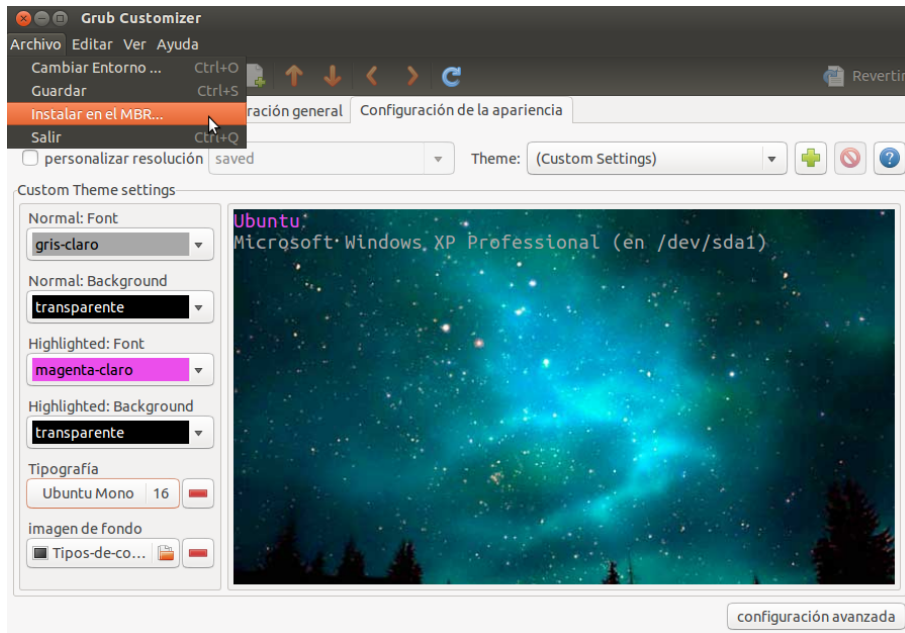
BOOTLOADER



BOOTLOADER



BOOTLOADER



6. Rescate: Se rompió el booteo

6.1. Reparar desde Consola

Antes de comenzar con esta parte es necesario aclarar este tipo de reparación esta apuntada a reinstalar un bootloader que ya estaba funcionando previamente en un sistema GNU/Linux ya instalado y configurado, que por algún motivo (MBR corrupto o pisado por el bootloader de Windows luego de su instalación) no esté funcionando. También para los mas osados que hayan estado jugando con `dd` o `fdisk` y se hayan mandado la parte. Primero vamos a establecer un contexto inicial para que usar de referencia en los ejemplos.

Tenemos una computadora, con un disco rígido SATA cableado como primario. Esto significa que Linux identificará este dispositivo como un archivo de nombre `sda` ubicado en el directorio `/dev/`, es decir `/dev/sda`.

Los archivos que se encuentran en el directorio `/dev/` son archivos de dispositivo (device files). Son especiales para los sistemas operativos UNIX, ya que representan y son el medio por el que se accede a los dispositivos de hardware, así como también a dispositivos virtuales (pseudo-devices). Generalmente los discos rígidos ubicados en puertos IDE se encuentran en los dispositivos de nombre `/dev/hdXY` donde X es la ubicación asignada según el cableado y la configuración de los jumpers ('a' para el primario del primer IDE, 'b' para el secundario del primer IDE, 'c' para el primario de segundo IDE y 'd' para el secundario del segundo IDE), e Y es el número de partición dentro de ese disco. Para los dispositivos SATA el nombre no será `/dev/hdXY` sino `/dev/sdXY`. Si no se especifica el número de partición, el archivo representa al disco completo.

El disco tiene 3 particiones, la primera es la partición del sistema que vino creada con la computadora cuando la compramos (`/dev/sda1`), la segunda es la partición donde tenemos instalado Windows (`/dev/sda2`) y la tercera donde tenemos instalado un Debian GNU/Linux 8.0 (Jessie) (`/dev/sda3`). Teníamos un GRUB ya configurado bien configurado, instalado en el MBR que hasta hace unas horas atrás nos permitía entrar en nuestro sistema GNU/Linux sin problemas, pero ahora acabamos de instalar Windows y el instalador pisó el MBR y borró el GRUB, impidiéndonos volver a entrar a Linux. Lo que queremos ahora es recuperar nuestro GRUB. Para esto vamos a tener que seguir una serie de pasos.

6.1.1. Consiguiendo las herramientas

Antes que nada necesitamos las herramientas para trabajar. Vamos a necesitar un CD de rescate o un live CD de GNU/Linux que nos permita como mínimo ejecutar comandos desde consola y montar una partición de nuestro disco en algún directorio y cambiar el directorio root con `chroot`.

Cabe aclarar que no todos los live CDs tienen las herramientas necesarias instaladas para realizar estas operaciones, por lo tanto hay que acostumbrarse a la prueba y el error. Además hay otros posibles problemas, como por ejemplo, que incluso habiendo podido montar la partición de nuestro SO no podamos ejecutar comandos desde el mismo porque el kernel y las librerías cargadas por el Live CD no sean compatibles con las compilaciones de los programas de nuestro SO. Por eso, siempre es bueno elegir un Live CD que sea de la misma distribución que el linux que venimos usando. Existe una herramienta llamada UnetBootin (<http://unetbootin.sourceforge.net/>) para Windows, Linux y MacOS que permite crear USBs booteables de diferentes distribuciones de Linux.

6.1.2. Booteo en modo de rescate

Con el CD o USB de rescate en mano, o mejor aún ya puesto en la computadora, vamos a encenderla y forzarla a bootear desde el mismo (asumimos que el lector sabe como hacer esto).

En el menú que aparece en primera instancia se presentarán varias opciones seguramente. Siempre queremos evitar las opciones que nos llevan directamente a la interfaz gráfica para evitar cualquier problema de incompatibilidades con placas de video. Este no es un problema común hoy en día, pero aún así es mejor prevenir que curar.

Una vez que el sistema haya booteado tendremos andando un SO son suficientes herramientas para hacer nuestra magia.

6.1.3. Planeando (y clarificando) la estrategia

Antes de ponernos a hacer cosas a ciegas es bueno tener cierta idea de lo que vamos a hacer, para que si nos topamos con algún error desafortunado no estemos tan perdidos o hasta podamos deducir porque se produce y corregirlo.

Lo que planeamos hacer a continuación es lo siguiente:

1. Vamos a montar las particiones del GNU/Linux que está instalado en nuestro disco en un directorio que decidamos , para poder acceder a sus contenidos.

En pocas palabras, en UNIX se le dice montar una partición a asociar los contenidos de una partición a un directorio.

Dado que UNIX tiene una estructura de directorios de arbol jerárquico en la que todo comienza desde la raíz (root) simbolizada con una barra (/) y cuelga de ella, los contenidos de las particiones no se visualizan como en windows, identificando a la misma con un símbolo (como disco C: o disco D:) sino que se

BOOTLOADER

monta la partición en un directorio, y para ver los contenidos de esa partición lo hacemos transparentemente como si se tratase de otro directorio cualquiera. Esta manera de manipular los filesystems es común a todos los sistemas UNIX incluido GNU/Linux

2. Luego vamos a ejecutar un shell (línea de comandos) al que le vamos a decir que tome como directorio de root el directorio que acabamos de montar en el paso anterior, para hacerle creer que los comandos que estamos ejecutando los estamos ejecutando desde el sistema operativo que tenemos instalado en el disco y no desde el CD de rescate.
3. Por último vamos a reinstalar el grub en el MBR.

6.1.4. Poniendo manos a la obra

A partir de aca el camino es fácil, teniendo la estrategia planeada y el contexto presentado, es solo cuestión de presentar los comandos para llevarlo a cabo.

Pero antes, una aclaración más: Lo más probable es que al haber arrancado el cd de rescate nos haya llevado a alguna especie de menú o interfaz que nos presente opciones o nos guíe en algún tipo de instalación. Nosotros no queremos hacer esto, solo necesitamos entrar en una línea de comandos. Así que lo que vamos a hacer es apretar las teclas ALT + F2 (o F3, o F4) hasta encontrar alguna terminal disponible en la cual podamos ejecutar comandos.

Hecha esta aclaración, MANOS A LA OBRA!

1. Ya que dijimos que nuestro GNU/Linux estaba instalada en la partición /dev/sda3 vamos a montarla:

```
mkdir /miLinux
mount /dev/sda3 /miLinux
```

Ahora hay que hacer un trquito extra: cuando el SO operativo inicia asocia los dispositivos físicos a device files (como lo mencionamos más arriba) que están ubicados en el directorio /dev/. El problema es que en el siguiente paso vamos a querer engañar al sistema operativo en ejecución corriendo una línea de comandos con su directorio de root cambiado a /miLinux pero si hacemos esto sin el truco que mostraremos ahora, el directorio que en ese momento va a funcionar como /dev/ (que en realidad va a ser /miLinux/dev, no va a tener asociados los device-files a los dispositivos).

Por lo tanto, lo que vamos a hacer es montar virtualmente el directorio /dev/ en /miLinux/dev. Si esta explicación no fue del todo clara pueden buscar información al respecto de looback devices y chroot en casa. Mientras tanto, con seguir con los pasos mencionados, todo debería funcionar.

BOOTLOADER

```
mount -o loop /dev /miLinux/dev
```

2. Ahora vamos a correr una línea de comandos diciéndole que el directorio /miLinux será su directorio de root.

```
chroot /miLinux
```

3. Y por último, vamos a ejecutar el instalador de grub diciéndole que se instale en el MBR del disco rígido.

```
grub-install /dev/sda
```

Voilà! Eso es todo. ahora podemos reiniciar la computadora y si todo salió bien, GRUB volverá a estar instalado igual que antes.

6.1.5. Consideraciones finales

Ahora solo hay un detalle a tener en cuenta, y es el mensaje de éxito que vino justo antes de esta sección: "GRUB volverá a estar instalado igual que antes". Esto quiere decir que si mi GRUB había sido sobre-escrito por una instalación de Windows (o sea, que acabamos de instalar Windows), el menú de GRUB no nos presentará la opción de bootear ese SO ya que en la configuración del GRUB que teníamos eso no estaba contemplado. Así que para agregarlo tienen que volver al punto 5 de este manual y agregarlo.

6.2. Reparar usando Boot-Repair

Boot-Repair es una herramienta que sirve para reparar errores en el sector de booteo de un disco, que pueden darse después de instalar un segundo sistema operativo o luego de alguna actualización fallida. Presenta una interfaz simple que permite solucionar la mayoría de los problemas en unos pocos pasos. También permite configuraciones avanzadas en caso de necesitar mayor control. Es software libre bajo licencia GNU-GPL.

6.2.1. Booteo

Instalar Boot-Repair-Disk [7] en una memoria USB y bootear. Se recomienda no quemarlo en un DVD si la computadora tiene Windows 8 pre-instalado o si bootea en modo EFI.

Bootear desde un CD de Ubuntu mediante la opción *Try Ubuntu without installing*. Luego, instalar Boot-Repair desde una terminal, con los siguientes comandos:

```
$ sudo add-apt-repository ppa:yannubuntu/boot-repair
$ sudo apt-get update
$ sudo apt-get install boot-repair
$ boot-repair
```

Es posible que durante el arranque el programa necesite actualizar o instalar algún paquete. También puede preguntar si existe un RAID de discos. Si no se sabe se debería indicar que no.

6.2.2. Reparación recomendada

Con el programa esté corriendo, hacer click en la opción *Recommended repair*. Durante la ejecución es posible que el programa nos indique que es necesario ejecutar determinados pasos, según cual sea el conflicto. Cuando la reparación haya finalizado el programa mostrará una URL del tipo `paste.ubuntu.com/XXXXX`. Accediendo a esa dirección se puede ver un detalle de lo sucedido. Es importante guardar la URL o el contenido para poder verificar errores en caso de que la reparación no haya sido exitosa. Si todo salió bien, luego de reiniciar se debería poder volver a bootear normalmente. Si hubo algún problema, va a ser necesario revisar el contenido la URL para analizar errores. Si no es posible detectar los problemas será necesario consultar en algún foro especializado.

6.2.3. Ver información del disco

Si la reparación falló y no se guardó la información contenida en la URL, o bien si se quiere analizar el estado del disco, el programa cuenta con una opción de chequeo sin reparación a través de la opción *Create a Bootinfo summary*. Al finalizar mostrará una URL para consultar el detalle.

6.2.4. Opciones avanzadas

Main options: Podemos indicar si reparar GRUB o MBR, setear el timeout e indicar si queremos reparar también el filesystem.

GRUB location: Si tenemos varios discos o particiones booteables, podemos identificar sobre qué GRUB trabajar.

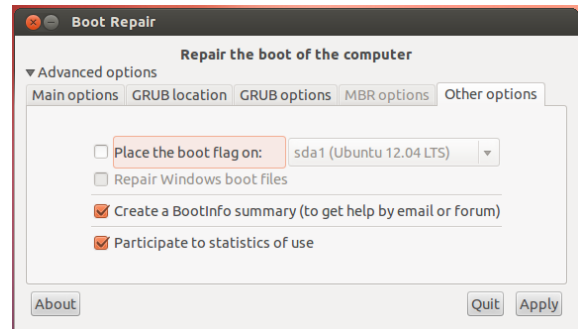
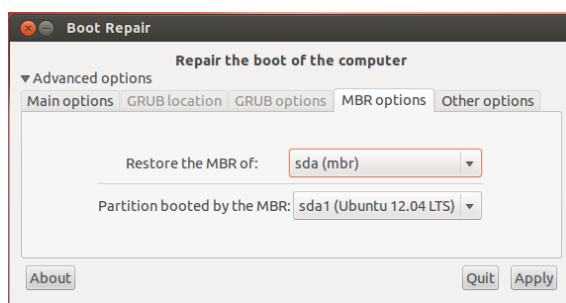
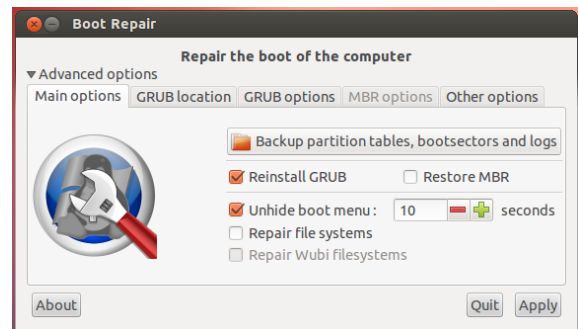
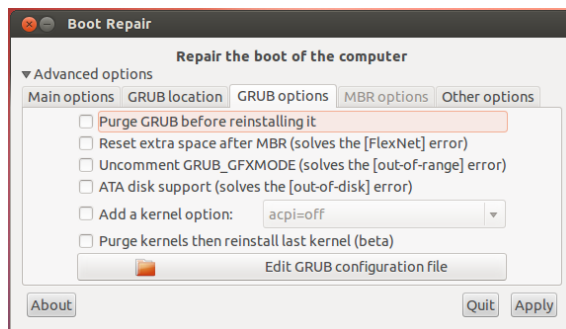
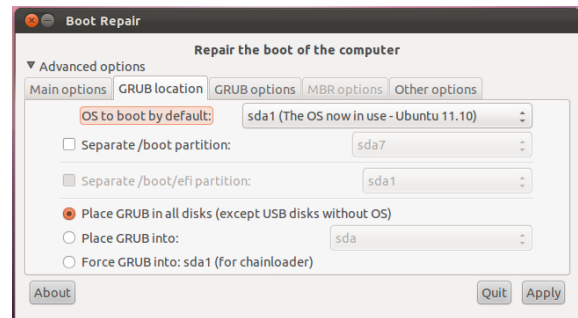
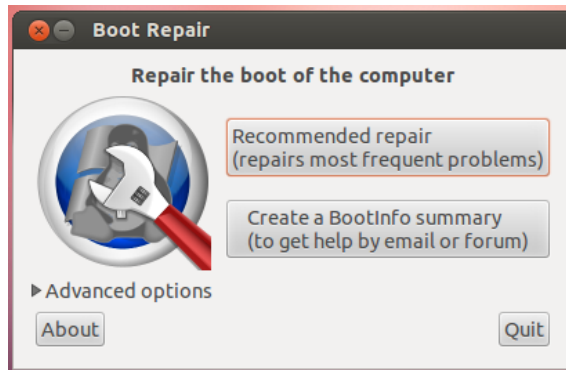
GRUB options: Diferentes opciones sobre GRUB

BOOTLOADER

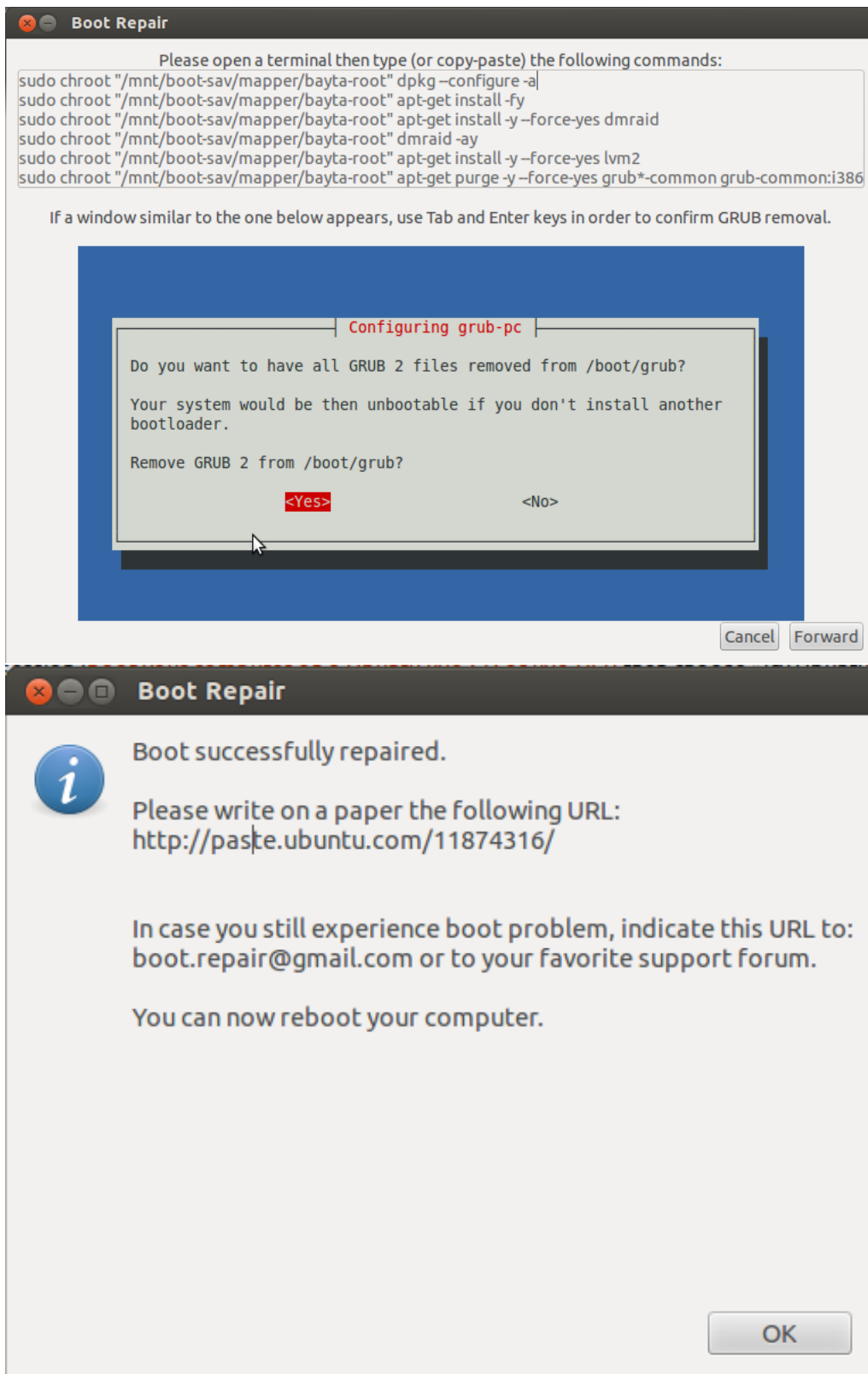
MBR options: Localizar la partición con el MBR a reparar.

Other options: Setear una partición como bootleable, generar o no el reporte, enviar información para estadística.

6.2.5. Screenshots



BOOTLOADER



7. Links útiles

Referencias

- [1] How computers boot
<http://duartes.org/gustavo/blog/post/how-computers-boot-up>
- [2] Kernel boot process
<http://duartes.org/gustavo/blog/post/kernel-boot-process/>
- [3] GRUB Documentation reference
<https://www.gnu.org/software/grub/grub-documentation.html>
- [4] GRUB2 bootloader - Full tutorial
<http://www.dedoimedo.com/computers/grub-2.html>
- [5] Help Ubuntu: Boot-Repair
<https://help.ubuntu.com/community/Boot-Repair>
- [6] Boot-Repair (oficial)
<https://sourceforge.net/p/boot-repair/home>
- [7] Boot-Repair-Disk
<http://sourceforge.net/p/boot-repair-cd/home/Home/>
- [8] Topic “Boot-repair: Graphical tool to repair the PC boot in 1 click !” - on Ubuntu forum
<http://ubuntuforums.org/showthread.php?p=10871917#post10871917>
- [9] HOWTO: easily create a Boot-Info summary
<https://help.ubuntu.com/community/Boot-Info>