Trocando o boot de BIOS para EFI

Para trocar o modo de boot de um sistema Linux (em particular, esse tutorial usa o Debian) de BIOS para EFI, seguiremos os seguintes passos:

- Troca do formato de particionamento de MBR para GPT
- Criando uma partição EFI
- Alteração do fstab para montar a partição EFI
- Instalação de um bootloader na partição EFI
- Instalação de uma entrada para o bootloader na EFI

Para seguir esse tutorial, utilize um Live CD de Linux (ou Ubuntu) com a mesma arquitetura que o sistema a ser modificado (para que seja possível usar o chroot). Se possível, utilize um CD que possa ser inicializado em modo UEFI, do contrário a etapa final de configuração da UEFI irá falhar e esse passo terá de ser realizado manualmente. Serão necessários os seguintes utilitários:

- gdisk (para converter a tabela de partição para GPT)
- gparted ou equivalente (para criar a partição EFI)
- efibootmgr (para adicionar a entrada do bootloader na EFI)

O tutorial usa como exemplo um sistema Debian. Para outras distribuições podem ser necessário ajustar:

- Instalação de pacotes (em particular para o grub-efi)
- Ponto de montagem da partição EFI

Troca do formato de particionamento de MBR para GPT

A troca do formato de particionamento pode ser feita com o comando gdisk. O gdisk deve ser executado como root e pede como argumento o caminho do dispositivo onde será feita a alteração, por exemplo

```
gdisk /dev/sda
```

Ao executar o programa, ele deve recolhecer a tabela MBR e avisar que fez uma conversão para GPT na memória, para então mostrar um prompt:

```
**************************************************************
Found invalid GPT and valid MBR; converting MBR to GPT format
in memory. THIS OPERATION IS POTENTIALLY DESTRUCTIVE! Exit by
typing 'q' if you don't want to convert your MBR partitions
to GPT format!
**************************************************************

Command (? for help):
```

O comando que nos é relevante é o w (write table to disk and exit). Execute-o e confirme a operação.

```
Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): Y
OK; writing new GUID partition table (GPT) to /dev/sda.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
```

Agora que a tabela foi convertida para gpt, precisamos criar uma partição EFI.

Criando uma partição EFI

No boot pela BIOS, o início do disco era chamado de MBR (Master Boot Record) e era onde se instalava o código para inicialização do sistema operacional (ou bootloader). Com o tempo, o tamanho da MBR passou a ser insuficiente para colocar o bootloader inteiro na MBR, resultando em esquemas que colocam apenas parte do código na MBR e o resto em uma das partições do disco.

No boot pela EFI, a ideia é que os bootloaders sejam instalados na EFI System Partition (ESP), que precisa seguir alguns padrões para ser reconhecida adequadamente pelo firmware. Precisamos criar essa partição essa partição manualmente.

Uma vez que o disco já está com a tabela convertida para gpt, podemos abrir o disco no gparted.

O tamanho da partição depende da necessidade. Macs costumam criar uma de 200 MiB, Windows apenas 100 MiB. Existem máquinas que tem problemas com partições menores que 512MiB. Por esse motivo, e por um detalhe na implementação do mkdosfs no linux, essa página recomenda um mínimo de 550MiB. Aqui criaremos uma partição de 1GB. Redimencione as partições para liberar o espaço desejado e ao criar a partição. O gparted permite já formatar a partição como FAT32. Aqui temos o mesmo disco com a partição já criada:

Para fazer com que a partição criada seja a partição EFI no gparted, clique com o botão direito na partição e selecione a opção Manage Flags. Na janela seguinte, selecione a flag esp, que também irá colocar a flag boot[1]:

Instalando um boot loader

Bootloader é o nome dado para o código que inicia o sistema operacional. Em sistemas que utilizam BIOS, este fica na MBR (Master Boot Record), que é um trecho de 512 bytes no início do disco e que

contem tanto código para iniciar o sistema quanto a tabela de partições. No linux em geral é instalado o GRUB como bootloader, que instala um código inicial nesse trecho, que então carrega o resto do código a partir de uma das partições do disco. Em sistemas EFI, como a especificação garante que o firmware é capaz de ler o formato FAT32, podemos instalar bootloaders como arquivos binários (que seguem o padrão da especificação) nessa partição, e adicionar entradas de boot na EFI que apontam diretamente para esses arquivos.

Existem diversas opções de bootloaders que podem ser utilizados para iniciar o Linux, como a versão EFI do GRUB, o rEFInd ou mesmo o embutido do kernel linux (Stub Loader). Aqui veremos como instalar o GRUB e usar o Stub Loader do kernel.

GRUB EFI

Para instalar corretamente o GRUB EFI, precisamos fazer um chroot no sistema. O procedimento é similar a recuperar um grub BIOS, a principal diferença sendo a necessidade de montar a partição EFI. Montamos então os caminhos necessários para o chroot:

```
mkdir /mnt/sda1
mount /dev/sda1 /mnt/sda1
mount -o bind /dev/ /mnt/sda1/dev
mount -o bind /sys/ /mnt/sda1/sys
mount -t proc /proc/ /mnt/sda1/proc
```

E montamos a partição EFI (o caminho pode variar para cada distro, /boot/efi é o utilizado no debian):

```
mkdir /mnt/sda1/boot/efi
mount /dev/sda2 /mnt/sda1/boot/efi
```

Antes de fazer o chroot, já vamos editar o fstab (que neste exemplo pode ser acessado pelo caminho /mnt/sda1/etc/fstab) e adicionar uma linha para a partição efi, como uma partição vfat comum:

```
UUID=9A9E-2FBF  /boot/efi        vfat    defaults    0    0
```

O UUID da partição pode ser determinado pelo comando blkid ou listando os symlinks em /dev/disk/by-uuid. Uma linha de comando que já adiciona a linha usando os caminhos dos exemplos acima é:

```
echo -e "UUID=$(blkid /dev/sda2 -s UUID -o
value)\t/boot/efi\t\tvfat\tdefaults\t0\t0" >> /mnt/sda1/etc/fstab
```

Podemos então fazer o chroot:

```
chroot /mnt/sda1
```

Uma vez dentro do sistema, podemos instalar o grub EFI. No debian, instalamos o pacote grub-efi:

```
apt-get install grub-efi
```

E instalamos o grub na partição EFI, especificando o caminho para o disco de destino:

```
grub-install /dev/sda
Installing for x86_64-efi platform.
Installation finished. No error reported.
```

Podemos confirmar que o grub se instalou corretamente na partição listando o conteúdo dela:

```
ls -R /boot/efi/
/boot/efi/:
EFI

/boot/efi/EFI:
debian

/boot/efi/EFI/debian:
grubx64.efi
```

E podemos verificar que ele se colocou na lista de inicialização da UEFI com o comando efibootmgr. A entrada 0000 agora é debian, e é a primeira da ordem de boot. Note que essa parte falha se a mídia live utilizada não for iniciada em modo EFI:

```
efibootmgr
BootCurrent: 0003
Timeout: 1 seconds
BootOrder: 0000,0003,0001,0002,0004,0005
Boot0000* debian
Boot0001* Hard Drive
Boot0002* CD/DVD Drive
Boot0003* UEFI: SanDisk
Boot0004* UEFI: SanDisk, Partition 1
Boot0005* USB
```

Instalando o kernel diretamente na partição EFi (EFI stub)

Atualmente, o kernel linux tem suporte a ser iniciado diretamente como um binário EFI, basta colocá-lo na partição EFI e adicionar a entrada adequada na lista de boot da EFI. No exemplo, o sistema está na partição /dev/sda1 e a partição EFI em /dev/sda2. Primeiramente montamos as partições:

```
mkdir /mnt/sda1
mount /dev/sda1 /mnt/sda1
mkdir /mnt/sda2
mount /dev/sda2 /mnt/sda2
```

Criamos então uma pasta para colocar o kernel. O nome é arbitrário, pode ser algo como 'debian', 'arch' ou similar. Em geral a pasta é criada dentro da pasta EFI da partição, mas mesmo isso não é obrigatório. Vamos criar uma pasta chamada 'banana' na raíz para ilustrar:

```
mkdir /mnt/sda2/banana
```

Então copiamos o kernel e o initrd do sistema para esta pasta. Aqui mudamos o nome do kernel apenas para vmlinuz e analogamente para a initrd a fim de simplificar a entrada na lista de boot da EFI, e permitir sobrescrever o kernel sem necessidade de refazê-la:

```
cp /mnt/sda1/boot/vmlinuz-3.16.0-4-amd64 /mnt/sda2/banana/vmlinuz
cp /mnt/sda1/boot/initrd.img-3.16.0-4-amd64 /mnt/sda2/banana/initrd.img
```

Criamos então a entrada na lista da EFI com o comando efibootmgr:

Adicionamos então a entrada na lista de boot da EFI:

```
efibootmgr -c -d /dev/sda -p 2 -L "Meu Linux sabor banana" -l
'\banana\vmlinuz' -u "root=/dev/sda1 ro initrd=\banana\initrd.img"
```

A partição do sistema também pode ser especificada pelo UUID:

```
export UUID=$(blkid -s UUID -o value /dev/sda1)
efibootmgr -c -d /dev/sda -p 2 -L "Meu Linux sabor banana" -l
'\banana\vmlinuz' -u "root=UUID=$UUID ro initrd=\banana\initrd.img"
```

Explicando melhor os parâmetros do comando:

- -c: Para criar uma entrada nova (já que o efibootmgr permite diversas operações)
- -d: O disco onde está a partição EFI a ser modificada.
- -p: A partição do disco que corresponde à partição EFI (no exemplo, colocamos 2 correspondendo a /dev/sda2)
- -L: O nome a ser exibido na lista de boot
- -l: O caminho para o loader. No nosso caso é o caminho direto para o kernel que colocamos na partição EFI (note que a barra precisa ser invertida).
- -u: Parâmetros adicionais a serem passados para o kernel. Note que os parâmetros são passados como uma única string, delimitada aqui por “ para não causar problema ao usar a variável $UUID.

A saída mostrará a lista de boot após a modificação. Aqui já havia uma entrada do grub com nome debian, e a entrada nova ficou como 0003:

```
BootCurrent: 0001
Timeout: 1 seconds
BootOrder: 0003,0000,0001,0002
Boot0000* debian
Boot0001* UEFI: SanDisk
Boot0002* UEFI: SanDisk, Partition 1
Boot0003* Meu Linux sabor banana
```

Manutenção do EFI stub

O procedimento acima coloca uma cópia do kernel atual na partição EFI e adiciona a entrada adequada para iniciar esse kernel. Quando o kernel é atualizado, essa cópia também precisa ser atualizada para iniciar o kernel novo. Existem vários várias formas de automatizar o processo [2)]. Aqui

utilizaremos o método da wiki do Debian[3], que consiste basicamente de criar um script no /etc/kernel/postinst.d que realiza o procedimento:

```
touch /etc/kernel/postinst.d/zz-update-efistub
chmod +x /etc/kernel/postinst.d/zz-update-efistub
```

No exemplo, o nome do arquivo começa com zz para que seja executado após os outros scripts (a ordem de execução dos scripts da pasta é alfabética). O conteúdo do arquivo (assumindo que o kernel e o initrd estão em /boot/efi/EFI/debian, sendo /boot/efi o ponto de montagem da partição EFI) é:

```
#!/bin/sh
cp /vmlinuz /initrd.img /boot/efi/EFI/debian/
```

Podemos extender o script para criar uma cópia do kernel atual por garantia:

```
#!/bin/sh
cp /boot/efi/EFI/debian/vmlinuz /boot/efi/EFI/debian/vmlinuz.old
cp /boot/efi/EFI/debian/initrd.img /boot/efi/EFI/debian/initrd.img.old
cp /vmlinuz /initrd.img /boot/efi/EFI/debian
```

Também podemos criar uma entrada para essa versão anterior do kernel, similar ao exemplo anterior:

```
export UUID=$(blkid -s UUID -o value /dev/sda1)
efibootmgr -c -d /dev/sda -p 2 -L "Debian (EFI stub) - previous kernel" -l
'\EFI\debian\vmlinuz.old' -u "root=UUID=$UUID ro
initrd=\EFI\debian\initrd.img.old"
```

Links Relevantes

- http://www.rodsbooks.com/efi-bootloaders/
- https://www.happyassassin.net/2014/01/25/uefi-boot-how-does-that-actually-work-then/
- http://www.uefi.org/specs/download
- https://wiki.debian.org/EFIStub
- https://wiki.archlinux.org/index.php/EFISTUB

1)
Essa “flag” é uma peculiaridade do gparted. Uma ESP é caracterizada pelo tipo de partição GUID C12A7328-F81F-11D2-BA4B-00A0C93EC93B, específico para esse fim, e pela flag de boot. A “flag” do gparted na verdade troca o tipo para esse além de marcar a flag de boot.

2)
A wiki do arch tem alguns exemplos: https://wiki.archlinux.org/index.php/EFISTUB

3)
https://wiki.debian.org/EFIStub

From:
https://wiki.ime.usp.br/ - **Wiki da Rede IME**

Permanent link:
https://wiki.ime.usp.br/tutoriais:trocando_o_boot_de_bios_para_uefi

Last update: **2024-03-19 13:23**