UEFI dual or more boot using rEFInd

Machines are now coming in with the UEFI standard. It presents some advantages like the ability to boot from disks over 2TiB and flexible pre-OS environments as drivers can be loaded in the UEFI firmware. But Linux users found that, suddenly, it was more difficult to multi-boot as we used to do in the past. However, Linux users have a variety of resources and we found a way to work with the new standard. It is not as straightforward as in the past, but it is not that hard either.

rEFInd is principally a tool to manage bootloaders, i.e. a boot manager. Some advantages of using rEFInd are

1. Manage and boot various different EFI binaries (.efi files),
2. Boot Linux kernels placed on the EFI partition; so rEFInd can also be a bootloader in itself,
3. There might be no need to install GRUB or ELILO onto EFI partition,
4. A GUI screen with multiple pre-defined options to boot EFI binaries or kernels at boot,
5. 😉 We can potentially have an ambitious multiboot for example with Windows orMac OS X, Slackware with ELILO and Ubuntu with GRUB2.

In this article, we will have a little summary of multi-booting on UEFI machines, going over the steps to make sure installation is done properly and how we can access our Linux distro(s); then we will describe the installation of rEFInd and how to use it. We will explain configuration of rEFInd carefully, as this will define the options and behaviour of our boot manager.

Installation of Linux distros

From now on, we will assume we have a hard disk using GPT with the following configuration:

- sda1 (FAT) - OEM software for security or any kind of maintainance software,
- sda2 (FAT) - EFI partition
- sda3 (NTFS) - Windows 10
- sufficiently unallocated space

Usually, we buy a machine and it comes with a hard disk having a huge partition with Windows or another OS on it. It is kind of bad practice to keep such a hard disk layout. So, we always advise people to partition their hard disks (PC or laptops) whenever they use it for the first time, whether they will dual boot or not. We shrink our main OS partition to an appropriate size and the rest of unallocated space can have partitions for our data. In this way, the OS is not hindered as the disk fills up with our personal files, so it continues to run and read the disk at a fairly high speed. Also, this avoids the risk of losing data if the OS crashes.

It is recommended to shrink the Windows partition from inside Windows itself using its disk manager and then do the other partitions with GPartEd or gdisk/cgdisk using any Linux bootable disk. Windows is notorious for placing files anywhere on a partition; so if you need to shrink it to a smaller space than it allowed, use a trusted third party software, or Windows might crash due to missing files and then you'll have to repair the Windows installation with a repair disk.

We will now proceed with our example. We assume we used gdisk and made the following partitions:

- sda4 - swap
- sda5 (Linux) - to install Slackware
- sda6 (Linux) - to install Ubuntu

We consider the installation of Slackware first, and then Ubuntu.

Slackware UEFI install

As from version 14.1 and onwards, Slackware64 supports EFI. It incorporates both ELILO and also an EFI-capable GRUB; you can choose between the two but ELILO is default.

ELILO

During the installation Slackware prompts you about installing ELILO to the EFI partition. If you choose to install it, a directory will be created on the EFI partition (EFI/Slackware) and the ELILO EFI binary (elilo.efi) will be placed there. The directory will also hold an elilo.conf file and a copy of the kernel image (vmlinuz). You should be able to boot into your Linux system right away.

GRUB

If ELILO was not installed then we need to initialize GRUB in order to have a bootable system. So, just after the installation process, do not reboot. We will chroot into our newly installed system right away.

Normally the newly installed system is mounted at /mnt. Use the **df** command to see what else is mounted and where.

→ Mount some useful filesystems (if they are not already present):

```
# mount -t vfat /dev/sda2 /mnt/boot/efi

# mount -t proc /proc /mnt/proc
# mount --rbind /sys /mnt/sys
# mount --rbind /dev /mnt/dev
# mount --rbind /run /mnt/run
```

→ chroot into newly installed system:

```
# chroot /mnt env -i HOME=/root TERM=$TERM PS1='\u@\h:\w# '
PATH=/usr/bin:/usr/sbin:/bin:/sbin bash --login +h
```

→ Then we install the GRUB bootloader on the EFI partition.

```
# modprobe efivars
# modprobe efivarfs
```

```
# modprobe dm-mod
# grub-install --target=x86_64-efi --efi-directory=/boot/efi --bootloader-
id=GRUB --recheck --debug

# mkdir -p /boot/grub/locale
# cp /usr/share/locale/en\@quot/LC_MESSAGES/grub.mo /boot/grub/locale/en.mo
# grub-mkconfig -o /boot/grub/grub.cfg
```

The modules efivars and efivarfs need to be loaded everytime before we use the **efibootmgr** command, here called by grub-install. Actually, it would have been loaded by grub-install automatically but we are mentioning this explicitly for completeness.
The grub-install command above generates lots of output. You can redirect it to a log file to check it out later. Among the last lines of the output we should observe the use of the **efibootmgr** command to register the EFI binary into the UEFI firmware settings. I strongly recommend reading the man pages for **efibootmgr** to understand the command.
One interesting thing here is that, in case Windows is installed, grub-mkconfig says it detects the Windows bootloader (bootmgfw.efi) but does not support it. So it will not be able to generate an entry for it in the grub.cfg. Then we should either put an entry manually in grub.cfg. This is something we can work around with rEFInd.

At this point we have the file EFI/GRUB/grubx64.efi and the /boot/grub tree. The grubx64.efi is registered in the firmware settings, so we can boot it. You can use the “-v” option with efibootmgr to see if everything is alright. If you need to change the order of some bootloaders, you can use the “-o” option. An example is shown below:

```
# efibootmgr -v
BootCurrent: 0000
Timeout: 0 seconds
BootOrder: 0001,3002,0000,2001,2002
Boot0000* Slackware     HD(2,145800,82000,366fa6fd-9e79-4904-a879-
fe0ef1ee349c)File(\EFI\Slackware\elilo.efi)
Boot0001* Windows Boot Manager  HD(2,145800,82000,366fa6fd-9e79-4904-a879-
fe0ef1ee349c)File(\EFI\Microsoft\Boot\bootmgfw.efi)
Boot2001* EFI USB Device        RC
Boot2002* EFI DVD/CDROM RC
Boot3002* Internal Hard Disk or Solid State Disk        RC

# efibootmgr -o 0000,0001,2001,2002,3002
# efibootmgr -v
BootCurrent: 0000
Timeout: 0 seconds
BootOrder: 0000,0001,2001,2002,3002
Boot0000* Slackware     HD(2,145800,82000,366fa6fd-9e79-4904-a879-
fe0ef1ee349c)File(\EFI\Slackware\elilo.efi)
Boot0001* Windows Boot Manager  HD(2,145800,82000,366fa6fd-9e79-4904-a879-
fe0ef1ee349c)File(\EFI\Microsoft\Boot\bootmgfw.efi)
Boot2001* EFI USB Device        RC
Boot2002* EFI DVD/CDROM RC
Boot3002* Internal Hard Disk or Solid State Disk        RC
```

Two more tips, in case you mess up anything ;-)

If you need to remove an entry from the list of EFI binaries, you can use the “-B” option. Let's say you want to remove EFI binary 0001:

```
efibootmgr -b 0001 -B
```

And if you need to add an EFI binary (let's say EFI/Slackware/elilo.efi) into the firmware settings:

```
efibootmgr -c -g -d /dev/sda -p 2 -w -L "ELILO" -l
'\EFI\Slackware\elilo.efi'
```

Just observe that our EFI partition is /dev/sda2; that is why there are “-d /dev/sda” and “-p 2”. These two arguments tell efibootmgr that our EFI partition is the 2nd partition on the device /dev/sda.

Ubuntu

Installation of Ubuntu is fairly straightforward. We boot into the DVD/USB and proceed with installation, since partitioning has already been done. Just be careful with Ubuntu where it asks where to install it. Ubuntu will detect Windows and will ask whether we want Ubuntu alongside, thus erasing the rest. So, be sure to stay in manual mode as much as possible. Choose “**Something else**” when prompted about the partitions. Also, make sure it detected the EFI partition.

At the end of installation, Ubuntu will have installed its GRUB on the EFI partition in directory EFI/ubuntu and it would have run **grub-mkconfig** and **efibootmgr**.

Installing rEFInd

For Slackware users, the easiest way to get rEFInd is to look at https://slackbuilds.org/. The SlackBuilds script repackages the pre-compiled binary for Slackware. After installing the package, we must go through a last step to have refind placed on the EFI partition. As root,

```
# /usr/share/refind-${VERSION}/install.sh
```

${VERSION} should be replaced with the version. This command will normally detect the EFI partition and system architecture, and copy the necessary files. It might prompt if it did not get some information automatically.

As for Ubuntu or other Linux distros, refind is most probably part of the official package repositories. Anyway, we only need to install refind on any one distro.

rEFInd Manual Install

Actually, the good thing with refind is that it does not have to be hosted on any OS. It can as well be copied directly to the EFI partition, provided you know which files to copy. So, you can download a refind archive from https://sourceforge.net/projects/refind/, and extract it. Normally, we would be using a 64-bit system, so create the directory /boot/efi/EFI/refind and copy the appropriate files there:

```
# mkdir /boot/efi/EFI/refind
# cp -R PATH_TO_REFIND_SOURCE/refind/drivers_x64 /boot/efi/EFI/refind/
# cp -R PATH_TO_REFIND_SOURCE/refind/icons /boot/efi/EFI/refind/
# cp -R PATH_TO_REFIND_SOURCE/refind/refind.conf-sample
/boot/efi/EFI/refind/refind.conf
# cp -R PATH_TO_REFIND_SOURCE/refind/refind_x64.efi /boot/efi/EFI/refind/
# cp -R PATH_TO_REFIND_SOURCE/keys /boot/efi/EFI/refind/
```

But, then we have to manually register our refind EFI binary into the firmware with efibootmgr:

```
# efibootmgr -c -g -d /dev/sda -p 2 -w -L "rEFInd" -l
'\EFI\refind\refind_x64.efi'
```

Again, our EFI partition is found on /dev/sda2.

Note that both ways of installing rEFInd can be done just after installation of Slackware (before rebooting). We just have to mount the necessary filesystems and chroot into the newly installed system. Then either install the refind Slackware package or manual install. There are many possibilities.

Configuring rEFInd

This is the most important part. Refind normally searches most directories on the EFI partition at boot and automatically detects all your EFI binaries or kernels, refind also scans volumes and will detect kernels, and lists them as options to boot from. But this is considered hazardous as it enables booting into Single User Mode or other recovery boot loaders. So, we will go for a manual configuration of the EFI/refind/refind.conf script for better control over the boot process.

rEFInd does not scan its own directory on the EFI partition (EFI/refind/).

If you boot into refind without modifying refind.conf, you'll see many boot entries on the start screen. We will modify refind.conf to remove much of these entries and place our custom entries there. The default refind.conf file placed by the install script at EFI/refind/ contains examples and some explanations.

Tidying up

Omit scanning volumes

We can disable the scanning of certain volumes by using the **dont_scan_volumes** command. We use the command with volume labels. The entry in refind.conf is

```
dont_scan_volumes "WINRE","Windows"
```

Omit scanning specific directories

We do not want refind to detect all the EFI binaries automatically, because then, we will not be able to control the number of entries. So we do not want refind to scan the directories where the EFI binaries are found. refind.conf already has the entry

```
dont_scan_dirs ESP:/EFI/boot,EFI/Dell,EFI/memtest86
```

We will add the following:

```
(we need to begin the list of directories with "+")
dont_scan_dirs + EFI/Boot,EFI/Microsoft,EFI/Slackware,EFI/ubuntu
```

Omit scanning specific files

This will ultimately avoid listing certain specific EFI binaries, since we will add a single entry for each one later.

```
dont_scan_files shim.efi,MokManager.efi,elilo-
x86_64.efi,bootmgfw.efi,bootmgr.efi
```

This will prevent listing of the ELILO from the Slackware installation or from any other directory on the EFI partition. And, it will omit the Windows binaries.

Omit scanning Linux kernels

```
scan_all_linux_kernels false
```

Scan directory for drivers

Normally, drivers should be placed in the tools/ directory on the EFI partition. Some drivers are found in refind's shared directory at /usr/share/refind-${VERSION}/refind/drivers_x64/. To enable scanning of EFI/tools:

```
scan_driver_dirs EFI/tools
```

Booting EFI binaries

Let us say we have the following directory tree on the EFI partition:

```
EFI/
 |
 |_ Boot/
 |     |_ bootx64.efi
 |
 |_ Microsoft/
 |     |_ Boot/
 |          |_ bootmgfw.efi
 |
 |_ Slackware/
 |     |_ elilo.efi
 |
 |_ ubuntu/
 |     |_ grubx64.efi
 |
 |_ refind/
       |_ refind_x64.efi
```

We should add custom entries for each of the EFI binaries we want to appear in the refind menu.

→ Entry for Windows:

```
menuentry Windows {
  icon EFI/refind/icons/os_win8.png
  loader EFI/Microsoft/Boot/bootmgfw.efi
}
```

Note that we can choose any of the icons from the EFI/refind/icons/ directory. Later we describe the use of custom made icons.

→ Entry for Slackware:

```
menuentry Slackware {
  icon EFI/refind/icons/os_win8.png
  loader EFI/Slackware/elilo.efi
}
```

→ Entry for Ubuntu:

```
menuentry Ubuntu {
  icon EFI/refind/icons/os_ubuntu.png
  volume 9f5b153d-d103-4314-bc98-455fa5d0c625
  loader EFI/ubuntu/grubx64.efi
}
```

Had to put the volume there as GRUB will have to access that volume for its configuration file.

Apple Mac

For a Mac, refind will automatically detect the EFI binary. It is most probably named “boot.efi”, so just make sure boot.efi is not in the list of files not to scan for. You might also need to place an hfs driver in the EFI/tools directory:

```
# cp /usr/share/refind-${VERSION}/refind/drivers_x64/hfs_x64.efi
/boot/efi/EFI/tools/
```

Booting Kernels

The nice part of refind is that we can also use it as a boot loader. So, we do not need GRUB or ELILO. We have to place the kernel we want to boot on the EFI partition. For this part we will use an example with the following directory tree under EFI/:

```
EFI/
 |_ Slackware14.1/
 |         |_ vmlinuz-generic-3.10.104
 |         |_ initrd-3.10.104.gz
 |
 |_ Slackware14.2/
 |         |_ vmlinuz-generic-4.4.29
 |         |_ initrd-4.4.29.gz
 |
 |_ Ubuntu16.04/
       |_ vmlinuz-4.4.16-21.generic
       |_ initrd.img-4.4.16-21.generic
       |_ vmlinuz-4.4.0-31.generic
       |_ initrd.img-4.4.0-31.generic
```

Note that neither Slackware nor Ubuntu place their generic kernels on the EFI partition by default. We made these directories manually and placed selected kernels there. A kernel (huge) is placed in directory EFI/Slackware only in the case where we install ELILO.

→ Entry for Slackware 14.1:

```
menuentry Slackware14.1_with_kernel_generic-3.10.104 {
  icon EFI/refind/icons/os_slackware.png
  volume cac2f895-6c8e-414c-8bc7-876519e828c0
  loader EFI/Slackware14.1/vmlinuz-generic-3.10.104
  options "ro root=UUID=cac2f895-6c8e-414c-8bc7-876519e828c0"
  initrd EFI/Slackware14.1/initrd-3.10.104.gz
}
```

The label of this menuentry is used as a short description and it is displayed at the bottom of the refind menu when hovering over the icons. The volume id for a particular filesystem can be obtained by using the **blkid** command:

```
# blkid /dev/sda12
/dev/sda12: UUID="9f5b153d-d103-4314-bc98-455fa5d0c625" TYPE="ext4"
```

→ Entry for Slackware 14.2:

```
menuentry "Slackware14.2 on sda12" {
  icon EFI/refind/icons/os_slackware.png
  volume 9f5b153d-d103-4314-bc98-455fa5d0c625
  loader EFI/Slackware14.2/vmlinuz-generic-4.4.29
  initrd EFI/Slackware14.2/initrd-4.4.29.gz
  submenuentry "Linux 4.4.29 generic" {
    options "ro root=UUID=9f5b153d-d103-4314-bc98-455fa5d0c625"
  }
  submenuentry "Linux 4.4.29 generic (recovery mode)" {
    options "ro root=UUID=9f5b153d-d103-4314-bc98-455fa5d0c625 single"
  }
}
```

This time we made an entry with submenu entries so that we have the option to boot into single user mode. Also note how we quoted the label because we used spaces. In the refind menu at boot, we will need to place the selection on this icon and press F2 in order to access the options. If ENTER is pressed right away on the icon, it will boot the default, that is the first submenu entry. The refind.conf entry could be made more concise using the **add_options** command. Since the only difference between the two options is the parameter “**single**”, we can just append it to the recovery mode's entry:

```
menuentry "Slackware14.2 on sda12" {
  icon EFI/refind/icons/os_slackware.png
  volume 9f5b153d-d103-4314-bc98-455fa5d0c625
  loader EFI/Slackware14.2/vmlinuz-generic-4.4.29
  initrd EFI/Slackware14.2/initrd-4.4.29.gz
  options "ro root=UUID=9f5b153d-d103-4314-bc98-455fa5d0c625"
  submenuentry "Linux 4.4.29 generic" {
  }
  submenuentry "Linux 4.4.14 generic (recovery mode)" {
    add_options "single"
  }
}
```

For Slackware users: we noticed that an initrd was needed even to boot the huge kernels because certain drivers are needed. In order to boot a huge kernel without an initrd, the drivers must be loaded from an EFI shell. Many drivers were placed in EFI/tools but when refind scanned the directory, it did not load all the drivers automatically. So, it is easier to simply make an initrd to accompany every kernel.

→ Entry for Ubuntu 16.04:

```
menuentry "Ubuntu 16.04" {
  icon EFI/refind/icons/os_ubuntu.png
  volume d3c5b9fa-53d7-42df-889f-78630cb9acea
  submenuentry "Linux 4.4.16-21 generic" {
```

```
    loader EFI/Ubuntu16.04/vmlinuz-4.4.16-21.generic
    options "ro root=UUID=d3c5b9fa-53d7-42df-889f-78630cb9acea"
    initrd EFI/Ubuntu16.04/initrd.img-4.4.16-21.generic
  }
  submenuentry "Linux 4.4.0-31 generic" {
    loader EFI/Ubuntu16.04/vmlinuz-4.4.0-31.generic
    options "ro root=UUID=d3c5b9fa-53d7-42df-889f-78630cb9acea"
    initrd EFI/Ubuntu16.04/initrd.img-4.4.0-31.generic
  }
}
```

Here we used submenu entries to have options between two different kernel versions. Further submenu entries could have been added in order to have single user modes, like above, for each kernel versions.

Using a refind_linux.conf file

If we have placed a kernel on the EFI partition, like above, we can also place a file called refind_linux.conf, holding the booting options, in the same directory as the kernel. For example,

→ EFI/Slackware14.2/refind_linux.conf :

```
"Boot with standard options" "ro root=UUID=9f5b153d-d103-4314-
bc98-455fa5d0c625"
"Boot into single-user mode" "ro root=UUID=9f5b153d-d103-4314-
bc98-455fa5d0c625 single"
"Boot with minimal options"  "root=UUID=9f5b153d-d103-4314-
bc98-455fa5d0c625"
```

→ Then the refind.conf entry for Slackware 14.2 becomes:

```
menuentry "Slackware14.2 on sda12" {
  icon EFI/refind/icons/os_slackware.png
  volume 9f5b153d-d103-4314-bc98-455fa5d0c625
  loader EFI/Slackware14.2/vmlinuz-generic-4.4.29
  initrd EFI/Slackware14.2/initrd-4.4.29.gz
}
```

The options will appear only after pressing F2 on the icon at the refind menu. ENTER on the icon will automatically boot the first set of options, i.e. “Boot with standard options”.

If we place two or more kernels in the same directory, then we need to have a separate entry like the above for each kernel. Then, the options from refind_linux.conf will be applied to each one.

Default selection and timeout

When the refind menu appears, if no keys are pressed before the timeout time, refind will boot the

default selection. If it is not defined explicitly in refind.conf, refind will boot the first entry on the screen. The timeout time is set with **timeout** and the default is set with **default_selection**.

```
timeout 10

default_selection 2  (will boot the second entry by default after timeout)

default_selection +  (will boot the loader most recently booted)

default_selection "Slackware14.1"  (will boot Slackware 14.1)
```

Concerning the last option, any substring that corresponds to the loader's title or volume will work. But we have to be careful when having similar entry titles like our two Slackware's above.

Non-bootloader tools

In the refind menu, below the icons for bootloaders and kernels selection, there is a set of icons for useful tools/options, e.g. EFI shell, memtest, partitioning program, about, shutdown, reboot. We can choose which options to be visible in the menu using the **showtools** command:

```
showtools reboot,shutdown,about
```

Options available are:

- shell : the EFI shell
- memtest : memtest86 program
- gdisk : partitioning software
- apple_recovery
- windows_recovery
- mok_tool : Machine Owner Key (MOK) maintainance tool
- shutdown
- reboot
- firmware : reboots the computer into the UEFI firmware settings
- about : information about rEFInd

Modifying Appearance of rEFInd

Now that we got our boot entries sorted out, we can play around and modify the appearance of the refind menu.

Resolution

We can set the resolution manually, to make sure that the banners/background will be well scaled. The system default resolution is usually 800×600.

```
resolution 1920 1080
```

Using custom icons

As we saw above with the custom menu entries we can choose icons for each entry with the **icon** option. So we can make our own icons and place them in directory EFI/refind/icons/. The default type of icons is bmp or png. There are two types of icons, small ones for the non-bootloader options (e.g. shutdown, reboot, …), and large ones for the bootloaders/kernel entries. All icons are square and default sizes are 48×48 for the small icons and 128×128 for the large icons. All icons must be more than 32×32 in size. The size of icons can be set using:

```
large_icon 128
small_icon 48
```

If the icons are physically smaller, they will be stretched to match the set size.

When the cursor is moved in the refind menu, a selection background switches from icon to icon. We can create our own selection backgrounds and copy them to the EFI/refind directory and then change refind.conf:

```
selection_big   myselectionbig.png
selection_small myselectionsmall.png
```

The formats accepted are again bmp and png. png is used mostly to have support for transparency. The default sizes are 64×64 for the small one and 144×144 for the big one.

And finally, to include our icon in our menu entry:

```
menuentry "Bootloader title" {
  icon EFI/refind/icons/mycustom_icon.png
  ...
  ...
}
```

Background

We can also create our own background and place it in the EFI/refind/ directory. The lines required in refind.conf are:

```
banner mybackground.bmp
banner_scale noscale
```

With **noscale** the image will be cropped if it is too large. The other possibility is **fillscreen**. Default is **noscale**.

Sources

- Originally written by aragorn2101

More information at:

- http://www.rodsbooks.com/refind/

howtos, uefi, efi, boot, slackware administration, author aragorn2101

From:
http://docs.slackware.com/ - **SlackDocs**

Permanent link:
http://docs.slackware.com/howtos:slackware_admin:uefi_triple_boot_with_refind_on_slackware

Last update: **2017/01/25 08:49 (GMT)**