Set up SYSLINUX as boot loader on UEFI based hardware

SYSLINUX is a boot loader that loads Linux (among other things) from FAT filesystem. The Syslinux Project, of which SYSLINUX the boot loader is a part, contains also ISOLINUX, EXTLINUX and PXELINUX, basically its variants to boot from optical media, ext2/3/4, Btrfs, XFS, UFS/FFS and NTFS or from network.

UEFI support seems to have been added in version 6 of SYSLINUX, while Slackware comes with version 4, which means it can not be set up during Slackware installation, unless one prepares SYSLINUX files on another machine.

Why SYSLINUX

That being said, why bother with SYSLINUX? A couple of reasons:

- LILO/ELILO is reportedly no longer maintained, the time may come we will have to switch to something else.
- Learning SYSLINUX may be worthwhile, because aside from serving as LILO/ELILO replacement, it can do many other things (e.g. PXE)
- Compared to GRUB, it is, well, less humongous and intimidating. After all, configuration files resemble those of LILO/ELILO.

Why UEFI

It is not as though there is some choice; machines nowadays *are* UEFI. One can, however, choose to set UEFI to so called “CSM” or sometimes “Legacy” mode. In this mode, UEFI acts as BIOS, i.e. reads the first sector (MBR) of a hard drive and loads and executes non-UEFI boot loader such as LILO from there. As opposed to UEFI in, say, UEFI mode, when it reads, loads and executes UEFI-aware boot loader from *EFI boot partition*.

Frankly, there are no compelling reasons not to use CSM (note that it is possible to use GPT partitions without UEFI). On the contrary, there are reasons to avoid UEFI booting, because:

- On some machines, it is not possible to turn off secure boot without also turning off UEFI booting (i.e. without turning on CSM).
- On some machines, it is possible to turn off secure boot, but anything but what came with the machine will not boot anyway.
- On some machines, UEFI tries really hard to boot Windows and ignores or resets boot order.
- On some machines, installing Linux in UEFI mode may brick the machine.

But, as with LILO, the day may come we will need to move on.

Installation

There are basically two steps: first, preferably before you even install Slackware, partition the disk using GPT and create EFI boot partition, second, compile and install SYSLINUX.

But before you begin:

- back up your data, if you have anything of value on any of drives in the machine
- boot to UEFI setup ("BIOS") and verify that boot method is set to "UEFI", not "CMS" or "Legacy"
- and also that Secure Boot is turned off

Create EFI Boot Partition

It is the place where UEFI will look for a boot loader. Several tools can be used to partition a drive and create EFI boot partition,:

- cfdisk (new versions; older do just DOS partitions)
- gdisk, an interactive tool and GPT counterpart to fdisk
- cgdisk, a menu-based tool and GPT counterpart to cfdisk
- sgdisk, apparently scriptable version of gdisk

Let's assume you have empty 10GB drive /dev/sda.

- Just in case, erase all previous MBR or GPT partitions.

```
# sgdisk -Z /dev/sda
```

- Verify the drive is clean.

```
# gdisk -l /dev/sda
GPT fdisk (gdisk) version 1.0.0

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries.
Disk /dev/sda: 20971520 sectors, 10.0 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): 8C026A41-F188-4521-A72C-D2B826EAA1D1
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 20971486
Partitions will be aligned on 2048-sector boundaries
Total free space is 20971453 sectors (10.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
```

- Partition the drive (using sgdisk or any above mentioned tool). Three partitions will be created, one for EFI, one for data and one for swap.

```
# sgdisk -n 1:0:+512M /dev/sda
# sgdisk -n 2:0:+9G /dev/sda
# sgdisk -n 3 /dev/sda
```

- Do not forget to set correct partition codes. EFI boot partitions has code 'ef00'.

```
# sgdisk -t 1:ef00 /dev/sda
# sgdisk -t 2:8300 /dev/sda
# sgdisk -t 3:8200 /dev/sda
```

- Verify layout.

```
# gdisk -l /dev/sda
GPT fdisk (gdisk) version 1.0.0

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sda: 20971520 sectors, 10.0 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): 3F2101FB-0D6B-481F-98B9-1C6621DB0981
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 20971486
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         1050623   512.0 MiB   EF00
   2         1050624        19924991   9.0 GiB     8300
   3        19924992        20971486   511.0 MiB   8200
```

- Create FAT32 filesystem on EFI boot partition.

```
# mkfs.vfat -F 32 /dev/sda1
```

That is pretty much it for EFI boot partition. You can do whatever you like with /dev/sda2 (RAID, LUKS, LVM, any filesystem) and install Slackware there, but leave EFI partition alone, that partition needs to be unencrypted and FAT32 formatted.

Set up SYSLINUX

Compilation

Download sources and compile.

```
$ unxz -c syslinux-6.03.tar.xz | tar -x
$ cd syslinux-6.03/
$ make
$ make INSTALLROOT=/some/where install
```

Set up boot loader

Now copy the boot loader and a few modules, as well as kernel and initrd to EFI boot partition created earlier. Let's asssume /dev/sda1 is mounted as /boot/efi.

- Copy the boot loader and a module for loading Linux. That is the bare minimum. And double check you are copying efi64 files.

```
# cd /boot/efi
# mkdir EFI
# mkdir EFI/SYSLINUX
# cd EFI/SYSLINUX
# cp /some/where/usr/share/syslinux/efi64/syslinux.efi .
# cp /some/where/usr/share/syslinux/efi64/ldlinux.e64 .
```

- Copy some other modules, e.g. for creating cute boot menu. For full list of modules and dependencies, see the list.

```
# cp /some/where/usr/share/syslinux/efi64/libutil.c32 .
# cp /some/where/usr/share/syslinux/efi64/menu.c32 .
```

- Create configuration file /boot/efi/EFI/SYSLINUX/syslinux.cfg. Bellow is an example of simple boot menu with two items.
    - `TIMEOUT 100` tells the boot loader to wait 100 hundredths of second, i.e. 10 seconds for user input.
    - If user selects nothing within 10 seconds, whatever is set as `DEFAULT` will be launched.
    - The first menu entry is Slackware 14.2.
        - `LABEL` is a name that can be used for `DEFAULT`.
        - `MENU LABEL` is a description.
        - `LINUX vmlinuz-huge-4.4.14` tells the boot loader to boot kernel vmlinuz-4.4.14 from the same directory (i.e. EFI\SYSLINUX\vmlinuz-4.4.14)
        - `APPEND /dev/vgroot/V_ROOT 4` is kernel command line telling the kernel to mount /dev/vgroot/V_ROOT as / (root). Obviously that is a logical volume in this case, but it could be e.g. plain /dev/sda2. The second argument, '4', will be in fact passed by kernel to init, which will start to runlevel 4 instead of default 3 (as per /etc/inittab), i.e. will start to graphical multi-user mode with X window system.
        - `INITRD initrd.gz-4.4.14` is an initrd image that is mounted as temporary / before the final / is mounted. Typically, an initrd is needed if kernel has to load some additional modules to access the root device, or has to perform some actions to make it accessible (think of RAID, LUKS, LVM). If you do not need initrd, simply delete this line.
    - The second menu entry is largely the same, except that there is no '4' in `APPEND`

/dev/vgroot/V_ROOT. Simply put, this is non-graphical safe mode. If X window system fails to start for whatever reason, you may end up with not only black screen, but also blocked keyboard and mouse and therefore unable to fix the cause. So it is a good idea to have this option prepared beforehand (another method to use otherwise unusable machine with broken X is to SSH-in, though it is a less reliable method).

/boot/efi/EFI/SYSLINUX/syslinux.cfg

```
UI menu.c32
PROMPT 0
TIMEOUT 100
DEFAULT Slackware 14.2

MENU TITLE Machine Boot Menu

LABEL Slackware 14.2
    MENU LABEL Slackware 14.2
    LINUX vmlinuz-huge-4.4.14
    APPEND /dev/vgroot/V_ROOT 4
    INITRD initrd.gz-4.4.14

LABEL Slackware 14.2 no X
    MENU LABEL Slackware 14.2 no X
    LINUX vmlinuz-huge-4.4.14
    APPEND /dev/vgroot/V_ROOT
    INITRD initrd.gz-4.4.14
```

- Still in /boot/efi/EFI/SYSLINUX, copy kernel here. Huge is the safer option. Generic may require initrd and more modules in it. If you do not know which they are, the machine might not boot, so go with the huge.

```
# cp /boot/vmlinuz-huge-4.4.14 .
```

- Create initrd if needed and copy here too. Man for options, but basically the command bellow creates an initrd that will contain some USB-related modules to enable keyboard input (because /dev/sda2 is encrypted (LUKS) and user will have to type in a passphrase).

```
# mkinitrd -c -k 4.4.14 -C /dev/sda2 -L -R -f xfs -m xfs:hid:hid-
generic:usbhid:xhci-hcd:xhci-pci:ehci-hcd:ehci-pci:ohci-hcd:ohci-pci:uhci-
hcd:uhci-pci:usb-storage -r /dev/vgroot/V_ROOT -o /boot/initrd.gz-4.4.14
# cp /boot/initrd.gz-4.4.14 .
```

- Finally, add SYSLINUX to UEFI boot menu. Note that there will be in fact two boot menus. The first is UEFI boot menu that will typically onyl show up if you press some key (e.g. F12 on Dell, F9 on HP, F11 with AsRock mainboard, often just Esc) and will present you with a list of devices or boot loaders to boot. Depending on how crappy your UEFI is, SYSLINUX may or may not show up there.

The second boot menu is that of SYSLINUX, i.e. what you typed to syslinux.cfg and will show up either if you do not tigger UEFI boot menu or trigger it and then select either SYSLINUX or device (hard drive) it is on.

```
# efibootmgr -c -d /dev/sda -p 1 -l \\EFI\\SYSLINUX\\syslinux.efi -L
"SYSLINUX"
Timeout: 2 seconds
BootOrder: 0000,0001,0002
Boot0000* SYSLINUX
Boot0001* UEFI: IP4 Realtek PCIe GBE Family Controller
Boot0002* UEFI: IP6 Realtek PCIe GBE Family Controller
```

- Syslinux has been added with code 0000 (Boot0000) and is 1st in boot order. If it was not, order would have to changed:

```
# efibootmgr -o 0000,0001,0002
```

- On some machines, it may be possible to change order also in UEFI (meaning its GUI/setup utility).

Troubleshooting

If your machine complains it has nothing to boot, that may indicate efibootmgr command was incorrect or somehow did not register. You may then try to force it to boot SYSLINUX by moving and renaming it to EFI\Boot\bootx64.efi, which is kind of fallback option; the machine will try to boot that at some point.

```
# cd /boot/efi/EFI
# mv SYSLINUX Boot
# cd Boot
# cp syslinux.efi bootx64.efi
```

Sources

- http://www.syslinux.org/wiki/index.php?title=The_Syslinux_Project
- https://wiki.archlinux.org/index.php/Unified_Extensible_Firmware_Interface

howtos, syslinux, uefi