How to boot CRUX with Syslinux

Author: Lin SiFuh

Overview

Syslinux is a lightweight bootloader for MS-DOS FAT filesystems (SYSLINUX), network booting (PXELINUX), bootable “El Torito” CD-ROMS (ISOLINUX), and Linux ext2/ext3/ext4 or btrfs filesystems (EXTLINUX).

This document details how to install and configure syslinux for DOS (MBR), GPT partition tables on legacy and UEFI systems running CRUX.

Installation

NOTE: It is important to know in advanced which type of system you will be installing on. Each installation, whether it be UEFI or legacy or if you are using DOS or GPT partition tables will be different.

UEFI installs will need efibootmgr which is included under the optional (opt) package set on the CRUX install CD. It is also important to note that even though UEFI does support DOS partition tables, some manufactures do not. So it is recommended to use GPT over DOS.

Setting up the partitions

To create a GPT partition table open fdisk and press the letter ‘g’. Alternatively you can press ‘o’ if you want a DOS partition table.

Once you have completed this step you can now partition your drives. You will want the first partition to be between 100MB to 550MB in size.

This partition must be set to bootable. For DOS partition tables you toggle the bootable flag by pressing the letter ‘a’. As for GPT you will need to press the letter ‘x’ to enter the expert menu and then ‘A’ to toggle LegacyBIOSBootable flag and ‘r’ to return back to the main menu again.

The partition type should have its Id set to ‘ef’ (EFI System or EFI (FAT-12/16/32)).

The filesystem can contain a FAT, ext2, ext3, ext4 or a BTRFS file system. I recommend using FAT as some UEFI systems may not support the other filesystems.

And finally the partition will be mounted as /boot.

If all goes well you can make the directory /mnt/boot and mount your newly created boot partition and proceed with your CRUX installation.

- Fdisk the device
- Change to GPT or DOS
- Create a partition between 100MB and 550MB in size.
- Change type to EFI System
- Toggle the bootable flag
- Format the filesystem
- Make the mount point
- And mount the boot partition

NOTE: These options were enabled in the kernel.

```
CONFIG_EFI_PARTITION=y

CONFIG_EFI=y

CONFIG_EFI_STUB=y

CONFIG_EFIVAR_FS=y

CONFIG_FB_EFI=y

CONFIG_EFI_VARS=y
```

Setting up, install and configuring syslinux

DOS(MBR)/GPT without UEFI

NOTE: For this system we will not be using UEFI. The boot partition is on /dev/sda1 and it is a FAT filesystem. The root partition is on /dev/sda2. Also the kernel has been compiled and copied into /boot

First we need to create the syslinux directory.

```
# mkdir /boot/syslinux
```

Then we copy the syslinux files needed.

```
# cp /usr/share/syslinux/*.c32 /boot/syslinux/
```

Now we install the syslinux bootloader on the boot partition. You can use the command **syslinux** for FAT systems or **extlinux** for ext2/ext3/ext4 or FAT systems. We will be using extlinux even though our partition is FAT.

```
# extlinux --install /boot/syslinux
```

Next we have to create the configuration file /boot/syslinux/syslinux.cfg and configure it for our system.

```
# vi /boot/syslinux/syslinux.cfg

        PROMPT 1
        TIMEOUT 10
        DEFAULT CRUX

                LABEL CRUX
                LINUX ../vmlinuz
                APPEND root=/dev/sda2 rw
```

Or if you are using an initrd or loading something like the intel-ucode, then your configuration might look something like this.

```
# vi /boot/syslinux/syslinux.cfg

        PROMPT 1
        TIMEOUT 10
        DEFAULT CRUX

                LABEL CRUX
                LINUX ../vmlinuz
                APPEND root=/dev/sda2 rw
                INITRD ../freestanding-00-intel-ucode.cpio,../freestanding-i915-firmware.cpio.xz
```

Do not leave INITRD with nothing following it. Either comment it out or remove the line completely as this will cause syslinux to fail during boot.

DOS(MBR)

To install the bootloader we use the command dd. This will write the syslinux master boot record to the disk we want to boot from. In our case /dev/sda is the disk we want to use.

```
# dd bs=440 count=1 conv=notrunc if=/usr/share/syslinux/mbr.bin of=/dev/sda
```

GPT

If your using a GPT disk then you will need to install the gptmbr.bin as opposed to the mbr.bin we used above.

```
# dd bs=440 count=1 conv=notrunc if=/usr/share/syslinux/gptmbr.bin of=/dev/sda
```

GPT and UEFI

NOTE: For this system we will be using UEFI, the boot partition is on /dev/sda1, it is a FAT filesystem and the root partition is on /dev/sda2. The kernel has been compiled and copied into /boot, but we will be changing this.

At the moment our kernel and initrds are in /boot. We will create the folder /boot/EFI and move our kernel into that directory. If you are using an initrd move them as well.

```
# mkdir /boot/EFI
# mv /boot/vmlinuz /boot/EFI/
```

Next we create the syslinux directory and copy the relevant files.

```
# mkdir /boot/EFI/syslinux
# cp /usr/share/syslinux/efi64/syslinux.efi /boot/EFI/syslinux
# cp /usr/share/syslinux/efi64/ldlinux.e64 /boot/EFI/syslinux
```

We then edit our configuration file exactly as we did above. The only difference is this that our configuration file is located under /boot/EFI/syslinux/ and not /boot/syslinux/.

```
# vi /boot/EFI/syslinux/syslinux.cfg

        PROMPT 1
        TIMEOUT 10
        DEFAULT CRUX

                LABEL CRUX
                LINUX ../vmlinuz
                APPEND root=/dev/sda2 rw
```

Or if you are using an initrd or loading something like the intel-ucode, then your configuration might look something like this.

```
# vi /boot/EFI/syslinux/syslinux.cfg

        PROMPT 1
        TIMEOUT 10
        DEFAULT CRUX

                LABEL CRUX
                LINUX ../vmlinuz
                APPEND root=/dev/sda2 rw
                INITRD ../freestanding-00-intel-ucode.cpio,../freestanding-i915-firmware.cpio.xz
```

Now we are going to install to UEFI. Assuming that you understand how UEFI works we are just going straight to the command. If you are unsure then I suggest you should do some further reading before continuing to the next step. https://crux.nu/Wiki/UEFI.

The command we will be using is efibootmgr. The device we are using is sda and the partition is partition 1. The name of the loader the UEFI will use is syslinux.efi and we will create the label CRUX

```
# efibootmgr -c -d /dev/sda -p 1 -l \\EFI\\syslinux\\syslinux.efi -L CRUX -v
```

That’s it! You should now have a fully bootable CRUX system with syslinux.

As a further note, syslinux gets all of its information from the configuration file syslinux.cfg. The only thing you need to do when modifying syslinux is edit the configuration file. You do not need to do anything else. Syslinux will read the configuration file upon the next boot cycle.

Switching between Legacy and UEFI mode in BIOS.

If you wish to be able to switch between legacy mode and UEFI in your BIOS but still be able to boot CRUX either way, then this can be achieved easily. Originally we installed legacy syslinux in the directory /boot/syslinux and the UEFI syslinux in /boot/EFI/syslinux. That means there are two configuration files. However this doesn’t have to be the case. We can actually place the UEFI install in other directories, including the directory used for the legacy install. Here is how.

```
# mkdir /boot/syslinux
# cp /usr/share/syslinux/*.c32 /boot/syslinux/
# cp /usr/share/syslinux/efi64/syslinux.efi /boot/syslinux
# cp /usr/share/syslinux/efi64/ldlinux.e64 /boot/syslinux
# extlinux --install /boot/syslinux
# vi /boot/syslinux/syslinux.cfg

        PROMPT 1
        TIMEOUT 10
        DEFAULT CRUX

                LABEL CRUX
                LINUX ../vmlinuz
                APPEND root=/dev/sda2 rw

# dd bs=440 count=1 conv=notrunc if=/usr/share/syslinux/gptmbr.bin of=/dev/sda
# efibootmgr -c -d /dev/sda -p 1 -l \\syslinux\\syslinux.efi -L CRUX -v
```

So what we did was create only /boot/syslinux. We copied all of the configuration files for legacy into /boot/syslinux and copied the efi loaders into the same directory. The configuration file was edited as normal and then after that we ran the command ‘dd’ which wrote the MBR to the bootable disk, /dev/sda in our case. Since UEFI doesn’t read the MBR and only reads the efi loader, we then ran the command ‘efibootmgr’ which told UEFI where the loader files are located. Neither installs conflict with each other. This will work with both GPT and MBR but as I mentioned above, even though UEFI does support DOS partition tables, some manufactures do not.