Syslinux

Syslinux is a collection of boot loaders capable of booting from drives, CDs, and over the network via **PXE**. Some of the supported **file systems** are **FAT**, **ext2**, **ext3**, **ext4**, and uncompressed single-device **Btrfs**.

Related articles

Arch boot process

Boot loaders

Warning: As of Syslinux 6.03, some of the features of the supported file systems are not supported by the bootloader; for example, the "64bit" feature of ext4 (boot) volumes. See **[1] (http://www.syslinux.org/wiki/index.php/Filesystem)** for more information.

Note: Syslinux, by itself, cannot access files from partitions other than its own. See **#Chainloading** on how to work around this.

Contents

BIOS Systems

Boot process overview

1. **Stage 1 : Part 1 - Load MBR** - At boot, the BIOS loads the 440 byte **MBR** boot code at the start of the disk (`/usr/lib/syslinux/bios/mbr.bin` or `/usr/lib/syslinux/bios/gptmbr.bin`).
2. **Stage 1 : Part 2 - Search active partition**. The **Stage 1 MBR boot code** looks for the partition that is marked as active (boot flag in MBR disks). Let us assume this is the `/boot` partition, for example.
3. **Stage 2 : Part 1 - Execute volume boot record** - The **Stage 1 MBR boot code** executes the Volume Boot Record (VBR) of the `/boot` partition. In the case of Syslinux, the VBR boot code is the starting sector of `/boot/syslinux/ldlinux.sys` which is created by the `extlinux --install` command. Note that `ldlinux.sys` is not the same as `ldlinux.c32` .
4. **Stage 2 : Part 2 - Execute `/boot/syslinux/ldlinux.sys`** - The VBR will load the rest of `/boot/syslinux/ldlinux.sys` . The sector location of `/boot/syslinux/ldlinux.sys` should not change, otherwise syslinux will not boot.

   Note: In the case of **Btrfs**, the above method will not work since files move around resulting in changing of the sector location of `ldlinux.sys` . Therefore, in Btrfs the entire `ldlinux.sys` code is embedded in the space following the VBR and is not installed at `/boot/syslinux/ldlinux.sys` unlike the case of other filesystems.

5. **Stage 3 - Load `/boot/syslinux/ldlinux.c32`** - The `/boot/syslinux/ldlinux.sys` will load the `/boot/syslinux/ldlinux.c32` (core module) that contains the rest of the **core** part of syslinux that could not be fit into `ldlinux.sys` (due to file-size constraints).

The `ldlinux.c32` file should be present in every Syslinux installation and should match the version of `ldlinux.sys` installed in the partition. Otherwise Syslinux will fail to boot. See **http://bugzilla.syslinux.org/show_bug.cgi?id=7** for more info.

6. **Stage 4 - Search and Load configuration file** - Once Syslinux is fully loaded, it looks for `/boot/syslinux/syslinux.cfg` (or `/boot/syslinux/extlinux.conf` in some cases) and loads it if it is found. If no configuration file is found, you will be dropped to a Syslinux `boot:` prompt. This step and the rest of **non-core** parts of Syslinux (`/boot/syslinux/*.c32` modules, excluding `lib*.c32` and `ldlinux.c32`) require `/boot/syslinux/lib*.c32` (library) modules to be present (**http://www.syslinux.org/wiki/index.php/Common_Problems#ELF**). The `lib*.c32` library modules and non-core `*.c32` modules should match the version of `ldlinux.sys` installed in the partition.

Installation on BIOS

Install the `syslinux (https://www.archlinux.org/packages/?name=syslinux)` package.

Note:

- `gptfdisk (https://www.archlinux.org/packages/?name=gptfdisk)` is required for **GPT** support using the automated script.
- If your boot partition is FAT, you will also need `mtools (https://www.archlinux.org/packages/?name=mtools)`.

Installing the package is not the same as installing the bootloader. After installing the relevant package(s), the bootloader code itself needs to be installed (to the adequate area, usually the VBR) so to be able to boot the system; the following sections provide alternative instructions depending on the characteristics of your particular system.

Automatic Install

Note: The `syslinux-install_update` script is Arch specific, and is not provided/supported by Syslinux upstream. Please direct any bug reports specific to the script to the Arch Bug Tracker and not upstream.

- After executing the `syslinux-install_update` script, do not forget to edit `/boot/syslinux/syslinux.cfg` by following **#Configuration** and **#Kernel parameters**.

Warning: The `syslinux-install_update` script sets a default root partition that possibly will not match your particular system. It is important to point Syslinux to the correct root partition by editing `/boot/syslinux/syslinux.cfg`, or the OS will fail to boot. See **#Kernel parameters**.

The `syslinux-install_update` script will install Syslinux, copy `*.c32` modules to `/boot/syslinux`, set the boot flag and install the boot code in the MBR. It can handle **MBR** and **GPT** disks along with software RAID:

If you use a separate boot partition, make sure that it is mounted. Check with `lsblk`; if you do not see a `/boot` mountpoint, mount it before you go any further.

- Run `syslinux-install_update` with flags: `-i` (install the files), `-a` (mark the partition *active* with the *boot* flag), `-m` (install the *MBR* boot code):

```
# syslinux-install_update -i -a -m
```

  If this command fails with *Syslinux BIOS install failed*, the problem is likely that the `extlinux` binary could not find the partition containing `/boot`:

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

```
extlinux: cannot find device for path /boot/syslinux
extlinux: cannot open device (null)
```

This can happen, for example, when upgrading from **LILO** which, while booting a current custom kernel, turned a kernel command line parameter of say `root=/dev/sda1` into its numeric equivalent `root=801`, as evidenced by `/proc/cmdline` and the output of the `mount` command. Remedy the situation by either continuing with the manual install described below while specifying `--device=/dev/sda1` to `extlinux`, or simply by first rebooting into a stock Arch Linux kernel; its use of an initramfs avoids the problem.

Note:

- If you rebooted your system now, you would get a Syslinux prompt. To automatically boot your system or get a boot menu, you need to create (edit) the configuration file.
- If you are on another root directory (e.g. from an install disk) install SYSLINUX by directing to the chroot:

```
# syslinux-install_update -i -a -m -c /mnt/
```

- Now is the time to edit `/boot/syslinux/syslinux.cfg` by following **#Configuration** and **#Kernel parameters**.

Manual install

Tip: If you are unsure of which partition table you are using (MBR or GPT), you can check using `blkid -s PTTYPE -o value /dev/sda`

Note: If you are trying to rescue an installed system with a live CD, be sure to **chroot** into it before executing these commands. If you do not chroot first, you must prepend all file paths (not `/dev/` paths) with the mount point.

Your boot partition, on which you plan to install Syslinux, must contain a FAT, ext2, ext3, ext4, or Btrfs file system. You do not have to install it on the root directory of a file system, e.g., with device `/dev/sda1` mounted on `/boot`. For example, you can install Syslinux in the `syslinux` subdirectory:

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

Copy all `.c32` files from `/usr/lib/syslinux/bios` to `/boot/syslinux` if you desire to use any menus or configurations other than a basic boot prompt. Do not symlink them.

```
# cp /usr/lib/syslinux/bios/*.c32 /boot/syslinux/
```

Now install the bootloader. For FAT, ext2/3/4, or btrfs boot partition use *extlinux*, where the device has been mounted:

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

Alternatively, for a FAT boot partition use *syslinux*, where the device is **unmounted**:

```
# syslinux --directory syslinux --install /dev/sda1
```

After this, proceed to install the Syslinux boot code (`mbr.bin` or `gptmbr.bin`) to the Master Boot Record 440-byte boot code region (not to be confused with MBR aka msdos partition table) of the disk, as described in the next sections, respectively.

Note: For a partitionless install, there is no need to install the Syslinux boot code to the MBR. You could skip below and jump to **#Configuration**. See **[2] (https://unix.stackexchange.com/questions/103501/boot-partiotionless-disk-with-syslinux)**.

MBR partition table

For an **MBR** partition table, ensure your boot partition is marked as "active" in your partition table (the "boot" flag is set). Applications capable of doing this include **fdisk** and **parted**. It should look like this:

```
# fdisk -l /dev/sda
```

```
[...]
  Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048      104447       51200   83  Linux
/dev/sda2          104448   625142447   312519000   83  Linux
```

Install the MBR:

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

An alternative MBR which Syslinux provides is: `altmbr.bin`. This MBR does *not* scan for bootable partitions; instead, the last byte of the MBR is set to a value indicating which partition to boot from. Here is an example of how `altmbr.bin` can be copied into position:

```
# printf '\x5' | cat /usr/lib/syslinux/bios/altmbr.bin - | dd bs=440 count=1 iflag=fullblock of=/dev/sda
```

In this case, a single byte of value 5 (hexadecimal) is appended to the contents of `altmbr.bin` and the resulting 440 bytes are written to the MBR on device `sda`. Syslinux was installed on the first logical partition (`/dev/sda5`) of the disk.

GUID partition table

For a **GPT**, ensure bit 2 of the attributes is set for the `/boot` partition using **gdisk**. In other words, the "legacy_boot" flag must be set. Using *sgdisk* the command is:

```
# sgdisk /dev/sda --attributes=1:set:2
```

This would toggle the attribute *legacy BIOS bootable* on partition 1 of `/dev/sda`. To check:

```
# sgdisk /dev/sda --attributes=1:show
```

```
1:2:1 (legacy BIOS bootable)
```

Install the MBR:

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

UEFI Systems

Note:

- `efi64` denotes x86_64 UEFI systems, for IA32 (32-bit) EFI replace `efi64` with `efi32` in the below commands.

- For Syslinux, the kernel and initramfs files need to be in the **EFI System Partition** (aka ESP), as Syslinux does not (currently) have the ability to access files outside its own partition (i.e. outside ESP in this case). For this reason, it is recommended to mount ESP at `/boot`.
- The automatic install script `/usr/bin/syslinux-install_update` does not support UEFI install.
- The configuration syntax of `syslinux.cfg` for UEFI is same as that of BIOS.

Limitations of UEFI Syslinux

- UEFI Syslinux application `syslinux.efi` cannot be signed by `sbsign` (from sbsigntool) for UEFI Secure Boot. Bug report: **[3] (http://bugzilla.syslinux.org/show_bug.cgi?id=8)**
- Using TAB to edit kernel parameters in UEFI Syslinux menu might lead to garbaged display (text on top of one another). Bug report: **[4] (http://bugzilla.syslinux.org/show_bug.cgi?id=9)**
- UEFI Syslinux does not support chainloading other EFI applications like `UEFI Shell` or `Windows Boot Manager`. Enhancement request: **[5] (http://bugzilla.syslinux.org/show_bug.cgi?id=17)**
- In some cases, UEFI Syslinux might not boot in some Virtual Machines like QEMU/OVMF or VirtualBox or some VMware products/versions and in some UEFI

emulation environments like DUET. A Syslinux contributor has confirmed no such issues present on VMware Workstation 10.0.2 and Syslinux-6.02 or later. Bug reports: **[6] (http://bugzilla.syslinux.org/show_bug.cgi?id=21)**, **[7] (http://bugzilla.syslinux.org/show_bug.cgi?id=23)** and **[8] (http://bugzilla.syslinux.org/show_bug.cgi?id=72)**
- Memdisk is not available for UEFI. Enhancement request: **[9] (http://bugzilla.syslinux.org/show_bug.cgi?id=30)**

Installation on UEFI

Note: In the commands related to UEFI, `esp` denotes the mountpoint of the **EFI System Partition** aka ESP.

- Install the **`syslinux (https://www.archlinux.org/packages/?name=syslinux)`** and **`efibootmgr (https://www.archlinux.org/packages/?name=efibootmgr)`** packages from the **official repositories**. Then setup Syslinux in the ESP as follows:

- Copy Syslinux files to ESP:

```
# mkdir -p esp/EFI/syslinux
# cp -r /usr/lib/syslinux/efi64/* esp/EFI/syslinux/
```

- Setup boot entry for Syslinux using **efibootmgr**:

```
# efibootmgr -c -d /dev/sdX -p Y -l /EFI/syslinux/syslinux.efi -L "Syslinux"
```

where `/dev/sdXY` is the partition containing the bootloader.

- Create or edit `esp/EFI/syslinux/syslinux.cfg` by following **#Configuration**.

Note: The config file for UEFI is `esp/EFI/syslinux/syslinux.cfg`, not `/boot/syslinux/syslinux.cfg`. Files in `/boot/syslinux/` are BIOS specific and not related to UEFI Syslinux.

Note: When booted in BIOS mode, **efibootmgr (https://www.archlinux.org/packages/?name=efibootmgr)** will not be able to set EFI nvram entry for `/EFI/syslinux/syslinux.efi`. To work around, place resources at the default EFI location: `esp/EFI/syslinux/* -> esp/EFI/BOOT/*` and `esp/EFI/syslinux/syslinux.efi -> esp/EFI/BOOT/bootx64.efi`

Configuration

The Syslinux configuration file, `syslinux.cfg`, should be created in the same directory where you installed Syslinux. In our case, `/boot/syslinux/` for BIOS systems and `esp/EFI/syslinux/` for UEFI systems.

The bootloader will look for either `syslinux.cfg` (preferred) or `extlinux.conf`

Tip:

- Instead of `LINUX`, the keyword `KERNEL` can also be used. `KERNEL` tries to detect the type of the file, while `LINUX` always expects a Linux kernel.
- `TIMEOUT` value is in units of **0.1 seconds**.

Examples

Note:

- Any configuration file found in the examples needs to be edited to set the proper kernel parameters. See section **#Kernel parameters**.
- Please, pay close attention to the paths. The examples may not be suitable for your installation, especially when using UEFI.
- The following examples assume that the kernel and initrd files are located one directory level up in relation to the location of `syslinux.cfg` (or, more precisely, one level up from the working directory).

Boot prompt

This is a simple configuration file that will show a `boot:` prompt and will automatically boot after 5 seconds. If you want to boot directly without seeing a prompt, set `PROMPT` to `0`.

Configuration:

```
* BIOS: /boot/syslinux/syslinux.cfg
* UEFI: esp/EFI/syslinux/syslinux.cfg
```

```
PROMPT 1
TIMEOUT 50
DEFAULT arch

LABEL arch
        LINUX ../vmlinuz-linux
        APPEND root=/dev/sda2 rw
        INITRD ../initramfs-linux.img

LABEL archfallback
        LINUX ../vmlinuz-linux
        APPEND root=/dev/sda2 rw
        INITRD ../initramfs-linux-fallback.img
```

Text boot menu

Syslinux also allows you to use a boot menu. To use it, copy the `menu` module to your Syslinux directory:

```
# cp /usr/lib/syslinux/bios/menu.c32 /boot/syslinux/
```

Copying additional `lib*.c32` library modules might be needed too.

Configuration:

```
* BIOS: /boot/syslinux/syslinux.cfg
* UEFI: esp/EFI/syslinux/syslinux.cfg
```

```
UI menu.c32
PROMPT 0
```

```
MENU TITLE Boot Menu
TIMEOUT 50
DEFAULT arch

LABEL arch
        MENU LABEL Arch Linux
        LINUX ../vmlinuz-linux
        APPEND root=/dev/sda2 rw
        INITRD ../initramfs-linux.img

LABEL archfallback
        MENU LABEL Arch Linux Fallback
        LINUX ../vmlinuz-linux
        APPEND root=/dev/sda2 rw
        INITRD ../initramfs-linux-fallback.img
```

For more details about the menu system, see **the Syslinux wiki (http://www.syslinux.org/wiki/index.php/Menu)**.

Graphical boot menu

Syslinux also allows you to use a graphical boot menu. To use it, copy the `vesamenu` COM32 module to your Syslinux folder:

```
# cp /usr/lib/syslinux/bios/vesamenu.c32 /boot/syslinux/
```

Copying additional `lib*.c32` library modules might be needed too.

Note: If you are using **UEFI**, make sure to copy from `/usr/lib/syslinux/efi64/` (`efi32` for i686 systems), otherwise you will be presented with a black screen. In that case, boot from a live medium and use **chroot** to make the appropriate changes.

This configuration uses the same menu design as the Arch Install CD, its config can be found at **projects.archlinux.org (https://projects.archlinux.org/archiso.git/tree/configs/releng/syslinux)**. The **Arch Linux background image (https://projects.archlinux.org/archiso.git/plain/configs/releng/syslinux/splash.png)** can be downloaded from there, too. Copy the image to `/boot/syslinux/splash.png`.

Configuration:

```
* BIOS: /boot/syslinux/syslinux.cfg
* UEFI: esp/EFI/syslinux/syslinux.cfg
```

```
UI vesamenu.c32
DEFAULT arch
PROMPT 0
MENU TITLE Boot Menu
MENU BACKGROUND splash.png
TIMEOUT 50

MENU WIDTH 78
MENU MARGIN 4
MENU ROWS 5
MENU VSHIFT 10
MENU TIMEOUTROW 13
MENU TABMSGROW 11
MENU CMDLINEROW 11
MENU HELPMSGROW 16
MENU HELPMSGENDROW 29

# Refer to http://www.syslinux.org/wiki/index.php/Comboot/menu.c32

MENU COLOR border       30;44   #40ffffff #a0000000 std
MENU COLOR title        1;36;44 #9033ccff #a0000000 std
MENU COLOR sel          7;37;40 #e0ffffff #20ffffff all
MENU COLOR unsel        37;44   #50ffffff #a0000000 std
MENU COLOR help         37;40   #c0ffffff #a0000000 std
MENU COLOR timeout_msg  37;40   #80ffffff #00000000 std
MENU COLOR timeout      1;37;40 #c0ffffff #00000000 std
MENU COLOR msg07        37;40   #90ffffff #a0000000 std
MENU COLOR tabmsg       31;40   #30ffffff #00000000 std


LABEL arch
```

```
        MENU LABEL Arch Linux
        LINUX ../vmlinuz-linux
        APPEND root=/dev/sda2 rw
        INITRD ../initramfs-linux.img


LABEL archfallback
        MENU LABEL Arch Linux Fallback
        LINUX ../vmlinuz-linux
        APPEND root=/dev/sda2 rw
        INITRD ../initramfs-linux-fallback.img
```

Since Syslinux 3.84, `vesamenu.c32` supports the `MENU RESOLUTION $WIDTH $HEIGHT` directive. To use it, insert `MENU RESOLUTION 1440 900` into your config for a 1440x900 resolution. However, the background picture has to have exactly the right resolution, as Syslinux will otherwise refuse to load the menu.

To center the menu and adjust resolution, use `MENU RESOLUTION`, `MENU HSHIFT $N` and `MENU VSHIFT $N` where `$N` is a positive number. The default values are both `0` which is the upper-left hand corner of your monitor. Conversely, a negative number starts from the opposite end of the screen (e.g. `VHSHIFT -4` would be 4 rows from the bottom of the screen).

To move the menu to the center, add or edit these values:

```
* BIOS: /boot/syslinux/syslinux.cfg
* UEFI: esp/EFI/syslinux/syslinux.cfg
```

```
MENU RESOLUTION 800 600 # or whatever your screen resolution is
MENU WIDTH 78           # width of the menu also required to bring the menu box to size
MENU VSHIFT 10          # moves menu down
MENU HSHIFT 10          # moves menu right
```

VESA standards are commonly a maximum of 25 rows and 80 columns, so going higher than those values might move the menu off the screen, potentially requiring editing from a rescue CD.

Kernel parameters

The **kernel parameters** are set by using the `APPEND` directive in `syslinux.cfg`: for each `LABEL` entry, a maximum of one **APPEND (http://www.syslinux.org/wiki/index.php/Config#APPEND)** line is accepted (i.e. spanning multiple lines is not valid).

It is recommended to make the following changes for the "fallback" entry as well.

In the simplest case, the partition name in the `root` parameter needs to be replaced. Change `/dev/sda2` to point to the correct root partition.

```
APPEND root=/dev/sda2
```

If you want to use UUID for **persistent block device naming** change the `APPEND` line as follows, substituting `1234` with the `UUID` of your root partition:

```
APPEND root=UUID=1234 rw
```

If you use encryption LUKS change the `APPEND` line to use your encrypted volume:

```
APPEND root=/dev/mapper/group-name cryptdevice=/dev/sda2:name rw
```

If you are using software RAID using mdadm (http://neil.brown.name/blog/mdadm), change the `APPEND` line to accommodate your RAID arrays. As an example the following accommodates three RAID 1 arrays and sets the appropriate one as root:

```
APPEND root=/dev/md1 rw md=0,/dev/sda2,/dev/sdb2 md=1,/dev/sda3,/dev/sdb3 md=2,/dev/sda4,/dev/sdb4
```

If booting from a software raid partition fails using the kernel device node method above an alternative, a more reliable, way is to use partition labels:

```
APPEND root=LABEL=THEROOTPARTITIONLABEL rw
```

If booting a btrfs subvolume, amend the `APPEND` line with `rootflags=subvol=<root subvolume>`. For example, where `/dev/sda2` has been mounted as a btrfs subvolume called 'ROOT' (e.g. `mount -o noatime,subvol=ROOT /dev/sda2 /mnt`), then the `APPEND` line would need to be modified as follows:

```
APPEND root=/dev/sda2 rw rootflags=subvol=ROOT
```

A failure to do so will otherwise result in the following error message:
`ERROR: Root device mounted successfully, but /sbin/init does not exist.`

Auto boot

If you do not want to see the Syslinux menu at all, use the **#Boot prompt**, and set `PROMPT` to `0` and comment out any `UI` menu entries. Setting the `TIMEOUT` variable to `0` might also be a good idea. Make sure there is a `DEFAULT` set in your `syslinux.cfg`. Holding either `Shift` or `Alt`, or setting either `Caps Lock` or `Scroll Lock`, during boot will allow for options other than default to be used. See the **upstream wiki (http://www.syslinux.org/wiki/index.php/Directives/special_keys)** for additional alternatives.

Security

Syslinux has two levels of bootloader security: a menu master password, and a per-menu-item password. In `syslinux.cfg`, use

```
MENU MASTER PASSWD passwd
```

to set a master bootloader password, and

```
MENU PASSWD passwd
```

within a `LABEL` block to password-protect individual boot items.

The passwd can be either a cleartext password or hashed: **see official documentation (http://www.syslinux.org/wiki/index.php/Comboot/menu.c32)**.

Chainloading

Note: Syslinux BIOS cannot directly chainload files located on other partitions; however, `chain.c32` can boot a partition boot sector (VBR).

If you want to chainload other operating systems (such as Windows) or boot loaders, copy the `chain.c32` module to the Syslinux directory (additional `lib*.c32` library modules might be needed too; for details, see the instructions in the previous section). Then create a section in the configuration file:

```
/boot/syslinux/syslinux.cfg
```

```
...
 LABEL windows
        MENU LABEL Windows
        COM32 chain.c32
        APPEND hd0 3
...
```

`hd0 3` is the third partition on the first BIOS drive - drives are counted from zero, but partitions are counted from one.

Note: For Windows, this skips the system's own boot manager (`bootmgr`), which is required for a few important updates (**eg. (http://support.microsoft.com/kb/2883200)**) to

complete. In such cases it may be advisable to temporarily set the MBR boot flag to the Windows partition (eg. with **GParted**), let the update finish installing, and then reset the flag to the Syslinux partition (eg. with Windows's own **DiskPart (http://www.online-tech-tips.com/computer-tips/set-active-partition-vista-xp)**).

If you are unsure about which drive your BIOS thinks is "first", you can instead use the MBR identifier, or if you are using GPT, the filesystem labels. To use the MBR identifier, run the command

```
# fdisk -l /dev/sdb
```

```
Disk /dev/sdb: 128.0 GB, 128035676160 bytes
255 heads, 63 sectors/track, 15566 cylinders, total 250069680 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xf00f1fd3

Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     4196351     2097152    7  HPFS/NTFS/exFAT
/dev/sdb2         4196352   250066943   122935296    7  HPFS/NTFS/exFAT
```

replacing `/dev/sdb` with the drive you wish to chainload. Using the hexadecimal number under Disk identifier: `0xf00f1fd3` in this case, the syntax in `syslinux.cfg` is

```
/boot/syslinux/syslinux.cfg
```

```
...
 LABEL windows
         MENU LABEL Windows
         COM32 chain.c32
         APPEND mbr:0xf00f1fd3
...
```

For more details about chainloading, see **the Syslinux wiki (http://www.syslinux.org/wiki/index.php/Comboot/chain.c32)**.

If you have **GRUB** installed on the same partition, you can chainload it by using:

/boot/syslinux/syslinux.cfg

```
...
 LABEL grub2
        MENU LABEL Grub2
        COM32 chain.c32
        append file=../grub/boot.img
...
```

Alternatively, it is also possible to load **GRUB** as a linux kernel by prepending `lnxboot.img` to `core.img`. The file `lnxboot.img` is part of `core/grub` and can be found in `/usr/lib/grub/i386-pc`.

/boot/syslinux/syslinux.cfg

```
...
 LABEL grub2lnx
        MENU LABEL Grub2 (lnxboot)
        LINUX ../grub/i386-pc/lnxboot.img
        INITRD ../grub/i386-pc/core.img
...
```

This may be required for booting from ISO images.

Chainloading other Linux systems

Chainloading another bootloader such as Windows' is pretty obvious, as there is a definite bootloader to chain to. But with Syslinux, it is only able to load files residing on the same partition as the configuration file. Thus, if you have another version of Linux on a separate partition, without a shared `/boot`, it becomes *necessary* to employ EXTLINUX rather than the other OS's default bootloader (eg. GRUB2). Essentially, EXTLINUX can be installed on the partition superblock/**VBR** and be called as a *separate bootloader* right from the MBR installed by Syslinux. EXTLINUX is part of The Syslinux Project and is included with the **`syslinux (https://www.archlinux.org/packages/?name=syslinux)`** package.

The following instructions assume you have Syslinux installed already. These instructions will also assume that the typical Arch Linux configuration path of `/boot/syslinux` is being used and the chainloaded system's `/` is on `/dev/sda3`.

From a booted Linux (likely the partition that Syslinux is set up to boot), mount the other system's root partition to your desired mount point. In this example this will be `/mnt`. Also, if a separate `/boot` partition is used on the second operating system, that will also need to be mounted. The example assumes this is `/dev/sda2`.

```
# mount /dev/sda3 /mnt
# mount /dev/sda2 /mnt/boot (only necessary for separate /boot)
```

Install EXTLINUX to the partition VBR, and copy necessary `*.c32` files

```
# extlinux -i /mnt/boot/syslinux/ (first create the directory if necessary)
# cp /usr/lib/syslinux/bios/*.c32 /mnt/boot/syslinux
```

Create `/mnt/boot/syslinux/syslinux.cfg`. You can use the other Linux's bootloader menu file for reference. Below is an example:

```
/mnt/boot/syslinux/syslinux.cfg on /dev/sda3
```

```
timeout 10

ui menu.c32

label OtherLinux
    linux /boot/vmlinuz-linux
    initrd /boot/initramfs-linux.img
    append root=/dev/sda3 rw quiet

label MAIN
    com32 chain.c32
    append hd0 0
```

And then add an entry to your main syslinux.cfg

```
/boot/syslinux/syslinux.cfg
```

```
label OtherLinux
    com32 chain.c32
    append hd0 3
```

taken from **Djgera's user wiki page**.

Note that the other Linux entry in `<other-OS>/boot/syslinux/syslinux.cfg` will need to be edited each time you update this OS's kernel unless it has symlinks to its latest kernel and initrd in /. Since we are booting the kernel directly and not chainloading the other-OS's default bootloader.

Using memtest

Install **memtest86+ (https://www.archlinux.org/packages/?name=memtest86%2B)** from the **official repositories**.

Use this `LABEL` section to launch **memtest**:

/boot/syslinux/syslinux.cfg

```
...
 LABEL memtest
        MENU LABEL Memtest86+
        LINUX ../memtest86+/memtest.bin
...
```

Note: If you are using PXELINUX, change the name from *memtest.bin* to *memtest* since PXELINUX treats the file with .bin extension as a boot sector and loads only 2KB of it.

HDT

HDT (Hardware Detection Tool) (http://hdt-project.org/) displays hardware information. Like before, the `.c32` file has to be copied from `/boot/syslinux/`. Additional `lib*.c32` library modules might be needed too. For PCI info, copy `/usr/share/hwdata/pci.ids` to `/boot/syslinux/pci.ids` and add the following to your configuration file:

/boot/syslinux/syslinux.cfg

```
LABEL hdt
        MENU LABEL Hardware Info
        COM32 hdt.c32
```

Reboot and power off

Note: As of Syslinux 6.03, `poweroff.c32` only works with APM and not with ACPI. For a possible solution, see **this thread (http://www.syslinux.org/archives/2012-March/017661.html)** .

Use the following sections to reboot or power off your machine:

/boot/syslinux/syslinux.cfg

```
LABEL reboot
        MENU LABEL Reboot
        COM32 reboot.c32

LABEL poweroff
        MENU LABEL Power Off
        COM32 poweroff.c32
```

Clear menu

To clear the screen when exiting the menu, add the following line:

/boot/syslinux/syslinux.cfg

```
MENU CLEAR
```

Keyboard layout

If you often have to edit your boot command with diverse parameters in the Syslinux boot prompt, then you might want to remap your keyboard layout. This allows you to enter "=", "/" and other characters easily on a non-US keyboard.

Note: keytab-lilo is a perl script invoking the "loadkeys" program.

To create a compatible keymap (e.g. a german one) run:

```
# keytab-lilo /usr/share/kbd/keymaps/i386/qwerty/us.map.gz /usr/share/kbd/keymaps/i386/qwertz/de.map.gz > /boot/syslinux/de.ktl
```

Now edit `syslinux.cfg` and add:

```
/boot/syslinux/syslinux.cfg
```

```
KBDMAP de.ktl
```

See the **Syslinux wiki (http://www.syslinux.org/wiki/index.php/Directives/kbdmap)** for more details.

Hiding the menu

Use the option:

```
/boot/syslinux/syslinux.cfg
```

```
MENU HIDDEN
```

to hide the menu while displaying only the timeout. Press any key to bring up the menu.

PXELINUX

Note: For UEFI, Syslinux uses the same binary for disk booting and network booting. Loading files from TFTP or other network protocols will require network booting Syslinux.

PXELINUX is provided by the **syslinux (https://www.archlinux.org/packages/?name=syslinux)** package.

For BIOS clients, copy the `{l,}pxelinux.0` bootloader to the boot directory of the client. For version 5.00 and newer, also copy `ldlinux.c32` from the same package:

```
# cp /usr/lib/syslinux/bios/pxelinux.0 "TFTP_root/boot"
# cp /usr/lib/syslinux/bios/ldlinux.c32 "TFTP_root/boot"
# mkdir "TFTP_root/boot/pxelinux.cfg"
```

We also created the `pxelinux.cfg` directory, which is where PXELINUX searches for configuration files by default. Because we do not want to discriminate between different host MACs, we then create the `default` configuration.

TFTP_root/boot/pxelinux.cfg/default

```
default linux

label linux
kernel vmlinuz-linux
append initrd=initramfs-linux.img quiet ip=:::::eth0:dhcp nfsroot=10.0.0.1:/arch
```

Or if you are using NBD, use the following append line:

```
append ro initrd=initramfs-linux.img ip=:::::eth0:dhcp nbd_host=10.0.0.1 nbd_name=arch root=/dev/nbd0
```

Note: You will need to change `nbd_host` and/or `nfsroot`, respectively, to match your network configuration (the address of the NFS/NBD server)

PXELINUX uses the same configuration syntax as SYSLINUX; refer to the upstream documentation for more information.

The kernel and initramfs will be transferred via TFTP, so the paths to those are going to be relative to the TFTP root. Otherwise, the root filesystem is going to be the NFS mount itself, so those are relative to the root of the NFS server.

To actually load PXELINUX, replace `filename "/grub/i386-pc/core.0";` in `/etc/dhcpd.conf` with `filename "/pxelinux.0"` (or with `filename "/lpxelinux.0"`).

Booting ISO9660 image files with memdisk

Syslinux supports booting from ISO images directly using the **memdisk (http://www.syslinux.org/wiki/index.php/MEMDISK)** module, see **Multiboot USB drive#Using Syslinux and memdisk** for examples.

Serial console

To enable Serial Console add the `SERIAL port [baudrate]` to the top of `syslinux.cfg` file. "port" is a number (0 for `/dev/ttyS0`), if "baudrate" is omitted, the baud rate default is 9600 bps. The serial parameters are hardcoded to 8 bits, no parity and 1 stop bit.**[10] (http://www.syslinux.org/wiki/index.php/SYSLINUX#SERIAL_port_.5Bbaudrate_.5Bflowcontrol.5D.5D)**

syslinux.cfg

```
SERIAL 0 115200
```

Enable Serial Console in the kernel at boot by adding `console=tty0 console=ttyS0,115200n8` to the `APPEND` option.**[11] (http://www.mjmwired.net/kernel/Documentation/kernel-parameters.txt#681)**

syslinux.cfg

```
APPEND root=UUID=126ca36d-c853-4f3a-9f46-cdd49d034ce4 rw console=tty0 console=ttyS0,115200n8
```

How to do this with GRUB: **Working with the serial console#GRUB2 and systemd**

Boot another OS once

It is possible to temporarily change the default Syslinux action and boot another label only during the next boot. The following command shows how to boot the `archfallback` label once:

```
# extlinux -o archfallback /boot/syslinux
```

During the next boot, the specified label will be booted without any Syslinux prompt showing up. The default Syslinux boot behaviour will be restored on the next reboot.

Troubleshooting

Failed to load ldlinux

An error message such as "Failed to load ldlinux.c32" during the initial boot can be triggered by many diverse reasons. One potential reason could be a change in file system tools or in a file system structure, depending on its own version. For instance, newer ext4 file systems might be created with its "64bit" feature enabled by default (whereas its "64bit" feature is only set manually, not by default, in older versions of mke2fs). This is just one example; file systems other than ext4 could also be affected by changes in their own structures and/or respective tools, thus also affecting bootloaders' behavior.

Warning: As of Syslinux 6.03, some of the features of the supported file systems are not supported by the bootloader; for example, the "64bit" feature of ext4 (boot) volumes. See **[12] (http://www.syslinux.org/wiki/index.php/Filesystem)** for more information.

Note: There is no direct and unique correspondence between a message such as `Failed to load ldlinux.c32` and a problem related to the file system:

- Other alternative symptoms, instead of this message, could also indicate a problem related to the file system.
- The message does not necessarily mean that the problem is related to the file system; there are other possible reasons for this type of messages.

See also **[13] (http://www.syslinux.org/wiki/index.php/Common_Problems#Failed_to_load_ldlinux)** (the whole page might be relevant for troubleshooting too).

Using the Syslinux prompt

You can type in the `LABEL` name of the entry that you want to boot (as per your `syslinux.cfg`). If you used the example configurations, just type:

```
boot: arch
```

If you get an error that the configuration file could not be loaded, you can pass your needed boot parameters, e.g.:

```
boot: ../vmlinuz-linux root=/dev/sda2 rw initrd=../initramfs-linux.img
```

If you do not have access to `boot:` in **ramfs**, and therefore temporarily unable to boot the kernel again,

1. Create a temporary directory, in order to mount your root partition (if it does not exist already):

```
# mkdir -p /new_root
```

2. Mount `/` under `/new_root` (in case `/boot/` is on the same partition, otherwise you will need to mount them both):

Note: Busybox cannot mount `/boot` if it is on its own ext2 partition.

```
# mount /dev/sd[a-z][1-9] /new_root
```

3. Use `vim` and edit `syslinux.cfg` again to suit your needs and save file.
4. Reboot.

Fsck fails on root partition

In the case of a badly corrupted root partition (in which the journal is damaged), in the ramfs emergency shell, mount the root file system:

```
# mount /dev/root partition /new_root
```

And grab the tune2fs binary from the root partition (it is not included in Syslinux):

```
# cp /new_root/sbin/tune2fs /sbin/
```

Follow the instructions at **ext2fs: no external journal** to create a new journal for the root partition.

No Default or UI found on some computers

Certain motherboard manufacturers have less compatibility for booting from USB devices than others. While an ext4 formatted USB drive may boot on a more recent computer, some computers may hang if the boot partition containing the *kernel* and *initrd* are not on a FAT16 partition. To prevent an older machine from loading `ldlinux` and failing to read `syslinux.cfg`, use `cfdisk` to create a FAT16 partition (<=2GB) and format using **dosfstools (https://www.archlinux.org/packages/?name=dosfstools)**:

```
# mkfs.msdos -F 16 /dev/sda1
```

then install and configure Syslinux.

Missing operating system

- Check that you have installed `gptmbr.bin` for GPT and `mbr.bin` for msdos partition table. A "Missing operating system" message comes from `mbr.bin` while `gptmbr.bin` would show a "Missing OS" message.
- Check whether the partition that contains `/boot` has the "boot" flag enabled.
- Check whether the first partition at the boot device starts at sector 1 rather than sector 63 or 2048. Check this with `fdisk -l`. If it starts at sector 1, you can move the partition(s) with `gparted` from a rescue disk. Or, if you have a separate boot partition, you can back up `/boot` with

```
# cp -a /boot /boot.bak
```

and then boot up with the Arch install disk. Next, use `cfdisk` to delete the `/boot` partition, and recreate it. This time it should begin at the proper sector, **63**. Now mount your partitions and `chroot` into your mounted system, as described in the beginners guide. Restore `/boot` with the command

```
# cp -a /boot.bak/* /boot
```

Check if `/etc/fstab` is correct, run:

```
# syslinux-install_update -iam
```

and reboot.

You will also get this error if you are trying to boot from a md **RAID** 1 array and created the array with a too new version of the metadata that Syslinux does not understand. As of August 2013 by default mdadm will create an array with version 1.2 metadata, but Syslinux does not understand metadata newer than 1.0. If this is the case you will need to recreate your **RAID** array using the `--metadata=1.0` flag to mdadm.

Windows boots up, ignoring Syslinux

Solution: Make sure the partition that contains `/boot` has the boot flag enabled. Also, make sure the boot flag is not enabled on the Windows partition. See the installation section above.

The MBR that comes with Syslinux looks for the first active partition that has the boot flag set. The Windows partition was likely found first and had the boot flag set. If you wanted, you could use the MBR that Windows or MS-DOS `fdisk` provides.

Menu entries do nothing

You select a menu entry and it does nothing, it just *"refreshes"* the menu. This usually means that you have an error in your `syslinux.cfg` file. Hit `Tab` to edit your boot parameters. Alternatively, press `Esc` and type in the `LABEL` of your boot entry (e.g. *arch*). Another cause could be that you do not have a kernel installed. Find a way to access your file system (through live CD, etc) and make sure that `/mount/vmlinuz-linux` exists and does not have a size of 0. If this is the case, **reinstall your kernel**[broken link: invalid section].

Cannot remove ldlinux.sys

The `ldlinux.sys` file has the immutable attribute set, which prevents it from being deleted or overwritten. This is because the sector location of the file must not change or else Syslinux has to be reinstalled. To remove it, run:

```
# chattr -i /boot/syslinux/ldlinux.sys
# rm /boot/syslinux/ldlinux.sys
```

White block in upper left corner when using vesamenu

Problem: *As of linux-3.0, the modesetting driver tries to keep the current contents of the screen after changing the resolution (at least it does so with my Intel, when having Syslinux in text mode). It seems that this goes wrong when combined with the vesamenu module in Syslinux (the white block is actually an attempt to keep the Syslinux menu, but the driver fails to capture the picture from vesa graphics mode).*

If you have a custom resolution and a `vesamenu` with early modesetting, try to append the following in `syslinux.cfg` to remove the white block and continue in graphics mode:

```
APPEND root=/dev/sda6 rw 5 vga=current quiet splash
```

Chainloading Windows does not work, when it is installed on another drive

If Windows is installed on a different drive than Arch and you have trouble chainloading it, try the following configuration:

```
LABEL Windows
      MENU LABEL Windows
      COM32 chain.c32
      APPEND mbr:0xdfc1ba9e swap
```

Replace the mbr code with the one your Windows drive has (details **above**), and append `swap` to the options.

Read bootloader log

In some cases (e.g. bootloader unable to boot kernel) it is highly desirable to get more information from the boot process. *Syslinux* prints error messages to screen but the boot menu quickly overwrites the text. To avoid losing the log information, disable `UI menu` in `syslinux.cfg` and use the default "command-line" prompt. It means:

- avoid the UI directive
- avoid ONTIMEOUT
- avoid ONERROR
- avoid MENU CLEAR
- use a higher TIMEOUT
- use PROMPT 1
- use DEFAULT <problematic_label>

To get more detailed debug log, **recompile** the **syslinux (https://www.archlinux.org/packages/?name=syslinux)** package with additional CFLAGS:

```
-DDEBUG_STDIO=1 -DCORE_DEBUG=1
```

Btrfs compression

Booting from btrfs with compression is not supported.**[14] (http://www.syslinux.org/wiki/index.php/Syslinux_4_Changelog#Changes_in_4.02)** This error will show:

```
btrfs: found compressed data, cannot continue!
invalid or corrupt kernel image.
```

Btrfs multi-device

Booting from multiple-device btrfs is not supported.[15] (http://repo.or.cz/syslinux.git/blob/HEAD:/extlinux/main.c) (As of 7/21/2016 line 1246 in validate_device_btrfs() in main.c) This head-scratching error will show (assuming you're installing on sda1):

```
/boot/syslinux is device /dev/sda1
extlinux: path /boot/syslinux doesn't match device /dev/sda1
```

See also

- Official website (http://www.syslinux.org)
- PXELinux configuration (http://www.josephn.net/scrapbook/pxelinux_stuff)
- Multiboot USB using Syslinux (http://blog.jak.me/2013/01/03/creating-a-multiboot-usb-stick-using-syslinux/)[dead link 2015-05-15]

Retrieved from "https://wiki.archlinux.org/index.php?title=Syslinux&oldid=510539"
