

Building an i386 Linux system for legacy BIOS using [Buildroot](#) involves configuring the target architecture and selecting a compatible bootloader like GRUB2 or Syslinux. [🔗](#)

## 1. Target Configuration

To target an i386 (32-bit x86) platform, start by configuring the core architecture in the Buildroot menu:

- Run `make menuconfig` in your Buildroot directory.
- Navigate to **Target options**.
- Set **Target Architecture** to `i386`.
- Set **Target Architecture Variant** to a specific CPU (e.g., `pentiumpro`) or leave it as the default for broad compatibility. [🔗](#)

## 2. Bootloader Selection (Legacy BIOS)

Legacy BIOS systems require a bootloader that can reside in the Master Boot Record (MBR). [🔗](#)

- **GRUB2**: Navigate to **Bootloaders > grub2**. Select the `i386-pc` platform, which is specifically designed for old BIOS boot methods.
- **Syslinux**: Alternatively, you can use Syslinux (specifically `isolinux` for CDs or `extlinux` for hard drives), which is a lightweight choice for BIOS systems. [🔗](#)

## 3. Creating a Bootable Image

Buildroot creates filesystem images (like `rootfs.ext2`), but these are not full disk images with partition tables. To create a bootable disk image (`.img`): [🔗](#)

1. **Enable Genimage**: Go to **System configuration** and ensure a tool like `genimage` is used to bundle the kernel and rootfs into a partitioned image.
2. **Manual Setup (Optional)**: If building manually, you must create a partition, copy the rootfs, and then install the bootloader using tools like `grub-bios-setup`.
3. **Output Files**: After running `make`, your bootable files (like `bzImage` and `rootfs.tar`) will be in `output/images/`. [🔗](#)

## 4. Testing in QEMU

You can verify your build without physical hardware using QEMU: [🔗](#)

```
bash
```

```
qemu-system-i386 -hda output/images/sdcard.img
```

Use code with caution.



If you only have a kernel and rootfs, you can boot them directly:

```
bash
```

```
qemu-system-i386 -kernel output/images/bzImage -append "root=/dev/sda" -h
```

Use code with caution.



The following tutorial demonstrates the process of configuring the kernel, building the system, and booting it in a virtual environment:



Making Linux Distro with Buildroot



Nir Lichtman  
YouTube • 30 Apr 2024

AI responses may include mistakes. [Learn more](#)



20 sites



Making Linux Distro with Buildroot

29 Apr 2024 — let's go ahead and demonstrate how you can make a simple Linux system using buildroot. so build routt is mostly used fo...



YouTube · Nir Lichtman



[Buildroot] Is grub-bios-setup the way to go to make a bootable ...

29 Jun 2014 — Partition it (either legacy or GPT style partitions work) > cfdisk disk.img > - Create one partition, type Linux, for t...



Buildroot

Trying to build a custom iso but it fails .. #263 - GitHub

16 Jul 2024 — IMPORTANT ! Check the directory in build options, change to the appropriate location of the buildroot you are actually ...



GitHub



Show all

thank you, but please help me with these files to  
make a bootable SD card for my old eeepc:...



To create a bootable SD card for a legacy BIOS EeePC using the Buildroot files, follow these steps to partition the card and install the Syslinux bootloader.

## 1. Partition the SD Card

Use `fdisk` to create a Master Boot Record (MBR) partition table with two partitions: [🔗](#)

1. **Boot Partition** (FAT32, ~100MB): For the kernel and Syslinux.
2. **Root Partition** (Ext2): For the `rootfs.ext2`.

bash

```
# Identify your SD card (e.g., /dev/sdX)
sudo fdisk /dev/sdX
# Actions in fdisk:
# 'o' (new MBR table)
# 'n' (new partition, primary, +100M) -> 't' (change type to 'b' for W95 /
# 'n' (new partition, primary, remaining space)
# 'a' (set partition 1 as bootable)
# 'w' (write and exit)
```

Use code with caution.



## 2. Format the Partitions

Format the first partition as FAT32 and the second as Ext2: [🔗](#)

bash

```
sudo mkfs.vfat -F 32 /dev/sdX1
sudo mkfs.ext2 /dev/sdX2
```

Use code with caution.



## 3. Install Syslinux Bootloader

Install the Syslinux boot code to the FAT32 partition: [🔗](#)

bash

```
sudo syslinux --install /dev/sdX1
```

Use code with caution.



Ensure you have the Syslinux MBR written to the head of the disk: [🔗](#)

bash

```
sudo dd if=/usr/lib/syslinux/mbr/mbr.bin of=/dev/sdX
```

Use code with caution.



## 4. Copy System Files

Mount the partitions and copy your Buildroot images: [↗](#)

### 1. Mounting:

```
bash
```

```
mkdir -p /mnt/boot /mnt/root  
sudo mount /dev/sdX1 /mnt/boot  
sudo mount /dev/sdX2 /mnt/root
```

Use code with caution.



2. **Kernel:** Copy `bzImage` (found in `output/images/`) to `/mnt/boot/`.

3. **Rootfs:** Since `rootfs.ext2` is a filesystem image, you can `dd` it directly to the partition or mount it and copy the contents.

1. *Direct Method:* `sudo dd if=output/images/rootfs.ext2 of=/dev/sdX2  
status=progress`

4. **Bootloader Config:** Create a file named `syslinux.cfg` in the root of `/mnt/boot/`:

```
text
```

```
DEFAULT buildroot  
LABEL buildroot  
  LINUX /bzImage  
  APPEND root=/dev/mmcblk0p2 rw rootwait console=ttyS0,115200 console=t
```

Use code with caution.



*Note: On an EeePC, the SD card is typically `/dev/mmcblk0`. If it fails to find the root, try `root=/dev/sda2`.* [↗](#)

## 5. Finalize

Unmount the partitions to ensure all data is written: [↗](#)

```
bash
```

```
sudo umount /mnt/boot /mnt/root
```

Use code with caution.



Insert the SD card into your EeePC, enter the BIOS (usually F2), and set the SD card (or USB Card Reader) as the primary boot device. [↗](#)

These guides detail how to partition, format, and install Syslinux to create a bootable SD card for an EeePC:

#### Trying to get syslinux to boot - OSDev.org

Sep 6, 2022 — 1. format a fresh drive with a fast32 partition and an ext2 partition. Make sure it's using...

 OSDev Forums 

#### How to create a bootable SD for the LS1043ARDB

Jun 25, 2021 — \* Please create two partitions with Linux command...

 NXP Community 



AI responses may include mistakes. [Learn more](#)



10 sites



#### Trying to get syslinux to boot - OSDev.org

5 Sept 2022 — 1. format a fresh drive with a fast32 partition and an ext2 partition. Make sure it's using mbr (I've also tried fat16 ...

 OSDev Forums 

#### How to create a bootable SD for the LS1043ARDB

24 Jun 2021 — \* Please create two partitions with Linux command "fdisk", one 1G FAT16 and 2G one ext2/ext3 or with remainder of the a...

 NXP Community 



#### How to put syslinux on an sd card

21 Jan 2013 — If a card reader appears in the OS as a generic USB storage device and is usable without installing any special drivers...

 Unix & Linux Stack Exchange 

Show all

was mache ich mit der isolinux.bin datei, die mir buildroot erstellt hat? sie ist 36kB groß

In deinem Fall ist die `isolinux.bin` vermutlich **nicht** die Datei, die du für eine SD-Karte benötigst.

Hier ist die Erklärung und die Lösung:

## 1. Das Problem: ISOLINUX vs. EXTLINUX

- **isolinux.bin** ist speziell für optische Medien (CDs/DVDs oder ISO-Dateien) gedacht. Sie sucht nach einer Konfiguration in einem Ordner namens `isolinux/`.

- **SD-Karten** (und Festplatten) nutzen im Syslinux-Paket normalerweise **extlinux** (für ext2/3/4 Partitionen) oder **syslinux** (für FAT-Partitionen).

Da du eine `rootfs.ext2` hast, ist der sauberste Weg für ein altes EeePC, die SD-Karte mit **EXTLINUX** bootbar zu machen.

## 2. So verwendest du deine Buildroot-Dateien richtig

Da du bereits eine `rootfs.ext2` hast, ist der einfachste Weg, diese Datei direkt auf die SD-Karte zu schreiben und dort den Bootloader zu installieren.

### Schritt-für-Schritt Anleitung:

#### 1. Image auf die SD-Karte schreiben:

Identifiziere deine SD-Karte (z.B. `/dev/sdb` , bitte genau prüfen!).

```
bash
```

```
sudo dd if=output/images/rootfs.ext2 of=/dev/sdX bs=1M status=progress
```

Use code with caution.



(Hinweis: Damit wird die gesamte SD-Karte zu einer ext2-Partition).

#### 2. Kernel kopieren:

Mount die SD-Karte, um den Kernel ( `bzImage` ) darauf zu speichern:

```
bash
```

```
sudo mount /dev/sdX /mnt  
sudo cp output/images/bzImage /mnt/boot/
```

Use code with caution.



#### 3. Bootloader installieren (statt isolinux):

Du brauchst das Tool `extlinux` (Teil des Syslinux-Pakets auf deinem Host-PC):

```
bash
```

```
sudo extlinux --install /mnt/boot/
```

Use code with caution.



#### 4. Konfiguration erstellen:

Erstelle eine Datei `/mnt/boot/extlinux.conf` (das ist das Äquivalent zur `isolinux.cfg`):

```
text
```

```
DEFAULT buildroot
LABEL buildroot
  LINUX /boot/bzImage
  APPEND root=/dev/mmcblk0 rw rootwait console=tty0
```

Use code with caution.



## 5. MBR schreiben (Wichtig für BIOS):

Damit das BIOS der EeePC weiß, dass es von der SD-Karte booten soll, muss der Master Boot Record geschrieben werden:

```
bash

sudo dd if=/usr/lib/syslinux/mbr/mbr.bin of=/dev/sdX
```

Use code with caution.



## Zusammenfassung zur `isolinux.bin`:

Du kannst die `isolinux.bin` ignorieren, sofern du kein `.iso` Image erstellen willst. Für deine SD-Karte ist sie technisch nicht geeignet, da sie nach dem ISO9660-Dateisystem sucht, während deine SD-Karte `ext2` nutzt.

### Tipp für die EeePC:

Wenn du in Buildroot unter `Target Packages > Hardware handling` die **Firmware für WLAN (z.B. Ath9k)** nicht mit eingebaut hast, wird das WLAN der EeePC später nicht funktionieren. Prüfe das ggf. nochmal in der `make menuconfig`.

AI responses may include mistakes. [Learn more](#)



buildroot hat mir leider kein bzImage gemacht. es liegt nicht in output/images/

Wenn Buildroot kein `bzImage` erstellt hat, liegt das meist daran, dass der Linux-Kernel in der Konfiguration noch nicht aktiviert wurde. Standardmäßig baut Buildroot oft nur das Dateisystem (`rootfs`), aber keinen Kernel.

Hier sind die Schritte, um das zu beheben:

## 1. Kernel in Buildroot aktivieren

1. Öffne die Konfiguration: `make menuconfig`
2. Navigiere zu **Kernel**.

3. Aktiviere das Feld **[X] Linux Kernel**.

4. Wähle unter **Kernel binary format** den Typ `bzImage` aus (dies ist der Standard für x86/i386).

5. Wähle eine **Kernel-Version**. Für ein altes EeePC empfiehlt sich "Same as toolchain" oder eine stabile Version wie "Latest (6.x.x)".

6. Wähle eine **Kernel-Konfiguration**. Für den Anfang ist `i386_defconfig` (unter *In-tree defconfig name*) eine gute Wahl für maximale Kompatibilität.

## 2. Bauvorgang erneut starten

Speichere die Konfiguration und führe den Build erneut aus:

```
bash
```

```
make
```

Use code with caution.



Buildroot wird nun den Kernel-Quellcode herunterladen, kompilieren und das `bzImage` nach `output/images/` kopieren.

## 3. Was tun mit `isolinux.bin` ?

Da du eine **SD-Karte** (keine CD) nutzt, ist die `isolinux.bin` für dich **nutzlos**.

- **Hintergrund:** `isolinux` ist nur für ISO9660-Dateisysteme (CDs).
- **Lösung:** Nutze stattdessen **extlinux** (für ext2/3/4) oder **syslinux** (für FAT), wie im vorherigen Schritt beschrieben. Diese Bootloader können das `bzImage` direkt von der SD-Karte laden.

## 4. Falls der Kernel-Build fehlschlägt

Sollte der Build mit einer Fehlermeldung abbrechen, fehlen oft Host-Pakete. Installiere zur Sicherheit diese Abhängigkeiten (auf einem Ubuntu/Debian Host):

```
bash
```

```
sudo apt install build-essential libncurses5-dev libssl-dev bc flex bison
```

Use code with caution.



Sobald der Build fertig ist, findest du die Datei hier:

```
output/images/bzImage
```

Diese Datei kopierst du dann in den `/boot/` Ordner deiner SD-Karte, wie oben beschrieben.