# Create a Custom Debian Live Environment (CD or USB)

Feb 8, 2018 *Updated Jun 27, 2020*

These are steps that I used on an **Ubuntu 20.04 LTS (Focal Fossa)** 64-bit system to build an **amd64 (64-bit) Debian 10 (Buster)** live environment that can boot from CD or USB.

The live environment generated by this guide is bootable with legacy BIOS or modern EFI hardware.

**See here for the previous version of the guide, which only used GRUB and was written for x86 (32-bit) Debian 9 using Ubuntu 18.**

▶ On the loss of 32-bit x86 support

I wrote this guide more for personal educational purposes than anything. It is not necessarily the fastest guide or the best guide for your needs. There are many other apps, tutorials, walkthroughs, and methods that better accomplish what is in this guide.[1][2][3][4][5][6][7] I hope this guide is helpful all the same.

The Debian DebianCustomCD documentation is more informative than this article could ever be.

**Warning**: I have **highlighted** all the places you should be in the **[chroot]** environment. Be careful! Running some of these commands on your local environment instead of in the chroot can damage your system.

## Prerequisites

Install applications we need to build the environment.

```
sudo apt install \
    debootstrap \
    squashfs-tools \
    xorriso \
    isolinux \
    syslinux-efi \
    grub-pc-bin \
    grub-efi-amd64-bin \
    mtools
```

Create a directory where we will store all of the files we create throughout this guide.

```
mkdir -p $HOME/LIVE_BOOT
```

## Bootstrap and Configure Debian

Set up the base Debian environment. I am using `buster` for my distribution and `amd64` for the architecture. Consult the list of debian mirrors.

*Please change the URL in this command if there is a mirror that is closer to you.*

```
sudo debootstrap \
    --arch=amd64 \
    --variant=minbase \
    buster \
    $HOME/LIVE_BOOT/chroot \
    http://ftp.us.debian.org/debian/
```

Chroot to the Debian environment we just bootstrapped.

```
sudo chroot $HOME/LIVE_BOOT/chroot
```

**[chroot]** Set a custom hostname for your Debian environment.

```
echo "debian-live" > /etc/hostname
```

**[chroot]** Figure out which Linux Kernel you want in the live environment.

```
apt-cache search linux-image
```

**[chroot]** I chose the image `linux-image-amd64`. I also believe `live-boot` is a necessity. `systemd-sys` (or an equivalent) is also necessary to provide init.

```
apt update && \
apt install --no-install-recommends \
    linux-image-amd64 \
    live-boot \
    systemd-sysv
```

**[chroot]** Install programs of your choosing, and then run `apt clean` to save some space. I use `--no-install-recommends` to avoid superfluous packages. You should decide what you need for your environment.

Read Debian's ReduceDebian article for tips on reducing the size of your Debian environment if size is important and you want a minimal and compact installation. Please note that some live environments like Alpine Linux, Tiny Core Linux, and Puppy Linux are specifically optimized for a tiny footprint.

Although this article results in a relatively tiny live environment, generating an environment that is only a couple dozen MB large takes additional effort not covered in this article.

```
apt install --no-install-recommends \
    network-manager net-tools wireless-tools wpagui \
    curl openssh-client \
    blackbox xserver-xorg-core xserver-xorg xinit xterm \
    nano && \
apt clean
```

**[chroot]** Set the root password. `root` will be the only user in this live environment by default, but you may add additional users as needed.

```
passwd root
```

**[chroot]** Exit the chroot.

```
exit
```

Create directories that will contain files for our live environment files and scratch files.

```
mkdir -p $HOME/LIVE_BOOT/{staging/{EFI/boot,boot/grub/x86_64-efi,isolinu
```

Compress the chroot environment into a Squash filesystem.

```
sudo mksquashfs \
    $HOME/LIVE_BOOT/chroot \
    $HOME/LIVE_BOOT/staging/live/filesystem.squashfs \
    -e boot
```

Copy the kernel and initramfs from inside the `chroot` to the `live` directory.

```
cp $HOME/LIVE_BOOT/chroot/boot/vmlinuz-* \
    $HOME/LIVE_BOOT/staging/live/vmlinuz && \
cp $HOME/LIVE_BOOT/chroot/boot/initrd.img-* \
    $HOME/LIVE_BOOT/staging/live/initrd
```

## Prepare Boot Loader Menus

Create an ISOLINUX (Syslinux) boot menu. This boot menu is used when booting in BIOS/legacy mode.

```
cat <<'EOF' >$HOME/LIVE_BOOT/staging/isolinux/isolinux.cfg
UI vesamenu.c32

MENU TITLE Boot Menu
DEFAULT linux
TIMEOUT 600
MENU RESOLUTION 640 480
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 linux
  MENU LABEL Debian Live [BIOS/ISOLINUX]
  MENU DEFAULT
  KERNEL /live/vmlinuz
  APPEND initrd=/live/initrd boot=live

LABEL linux
  MENU LABEL Debian Live [BIOS/ISOLINUX] (nomodeset)
  MENU DEFAULT
  KERNEL /live/vmlinuz
  APPEND initrd=/live/initrd boot=live nomodeset
EOF
```

Create a second, similar, boot menu for GRUB. This boot menu is used when booting in EFI/UEFI mode.

```
cat <<'EOF' >$HOME/LIVE_BOOT/staging/boot/grub/grub.cfg
search --set=root --file /DEBIAN_CUSTOM

set default="0"
set timeout=30

# If X has issues finding screens, experiment with/without nomodeset.

menuentry "Debian Live [EFI/GRUB]" {
    linux ($root)/live/vmlinuz boot=live
    initrd ($root)/live/initrd
}

menuentry "Debian Live [EFI/GRUB] (nomodeset)" {
    linux ($root)/live/vmlinuz boot=live nomodeset
    initrd ($root)/live/initrd
}
EOF
```

Create a third boot config. This config will be an early configuration file that is embedded inside GRUB in the EFI partition. This finds the root and loads the GRUB config from there.

```
cat <<'EOF' >$HOME/LIVE_BOOT/tmp/grub-standalone.cfg
```

```
search --set=root --file /DEBIAN_CUSTOM
set prefix=($root)/boot/grub/
configfile /boot/grub/grub.cfg
EOF
```

▶ Why both GRUB and SYSLINUX?

▶ Why grub.cfg and grub-standalone.cfg?

Create a special file in `staging` named `DEBIAN_CUSTOM`. This file will be used to help GRUB figure out which device contains our live filesystem. This file name must be unique and must match the file name in our `grub.cfg` config.

```
touch $HOME/LIVE_BOOT/staging/DEBIAN_CUSTOM
```

Your `LIVE_BOOT` directory should now roughly look like this.

```
LIVE_BOOT/chroot/*tons of chroot files*
LIVE_BOOT/staging/live/initrd
LIVE_BOOT/staging/live/vmlinuz
LIVE_BOOT/staging/live/filesystem.squashfs
LIVE_BOOT/staging/isolinux/isolinux.cfg
LIVE_BOOT/staging/EFI/boot
LIVE_BOOT/staging/boot/grub/grub.cfg
LIVE_BOOT/staging/boot/grub/x86_64-efi
LIVE_BOOT/staging/DEBIAN_CUSTOM
```

## Prepare Boot Loader Files

Copy BIOS/legacy boot required files into our workspace.

```
cp /usr/lib/ISOLINUX/isolinux.bin "${HOME}/LIVE_BOOT/staging/isolinux/"
cp /usr/lib/syslinux/modules/bios/* "${HOME}/LIVE_BOOT/staging/isolinux/
```

Copy EFI/modern boot required files into our workspace.

```
cp -r /usr/lib/grub/x86_64-efi/* "${HOME}/LIVE_BOOT/staging/boot/grub/x8
```

Generate an EFI bootable GRUB image.

```
grub-mkstandalone \
    --format=x86_64-efi \
    --output=$HOME/LIVE_BOOT/tmp/bootx64.efi \
    --locales="" \
    --fonts="" \
    "boot/grub/grub.cfg=$HOME/LIVE_BOOT/tmp/grub-standalone.cfg"
```

Create a FAT16 UEFI boot disk image containing the EFI bootloader. Note the use of the `mmd` and `mcopy` commands to copy our UEFI boot loader named `bootx64.efi`.

```
(cd $HOME/LIVE_BOOT/staging/EFI/boot && \
    dd if=/dev/zero of=efiboot.img bs=1M count=20 && \
    mkfs.vfat efiboot.img && \
    mmd -i efiboot.img efi efi/boot && \
    mcopy -vi efiboot.img $HOME/LIVE_BOOT/tmp/bootx64.efi ::efi/boot/
)
```

## Create Bootable ISO/CD

The `.iso` file we create can be burned to a CD-ROM (optical media) and can also be written to a USB device with `dd`. The process here a bit complex, but that resultant behavior is common in many modern live environments such as the Ubuntu installation `.iso` file.

*Please note that writing an `.iso` file to a USB device is not the same as installing the live environment directly to a USB device.*

Generate the `.iso` disc image file.

```
xorriso \
    -as mkisofs \
    -iso-level 3 \
    -o "${HOME}/LIVE_BOOT/debian-custom.iso" \
    -full-iso9660-filenames \
    -volid "DEBIAN_CUSTOM" \
    -isohybrid-mbr /usr/lib/ISOLINUX/isohdpfx.bin \
    -eltorito-boot \
        isolinux/isolinux.bin \
        -no-emul-boot \
        -boot-load-size 4 \
        -boot-info-table \
        --eltorito-catalog isolinux/isolinux.cat \
    -eltorito-alt-boot \
        -e /EFI/boot/efiboot.img \
        -no-emul-boot \
        -isohybrid-gpt-basdat \
    -append_partition 2 0xef ${HOME}/LIVE_BOOT/staging/EFI/boot/efiboot.
    "${HOME}/LIVE_BOOT/staging"
```

▶ Using UNetbootin, YUMI, or similar tools to install the ISO to USB

## Citations

- [syslinux] Isohybrid wiki page and UEFI
  - This makes me think syslinux EFI with isohybird will not work!
- [syslinux] Is efiboot.img required?
  - See: `This is what i understand from SYSLINUX being unable to boot from ISO 9660 via UEFI.
- Isohybrid - Syslinux Wiki
  - Note that it is not clearly expliained *how* we generate `efiboot.img`. Every major distro uses Grub *and* Syslinux for legacy + EFI booting.
- [SOLVED] Trying to make Syslinux visually pleasing... / Newbie Corner / Arch Linux Forums
  - Decent SYSLINUX vesa menu colors.
- GRUB/Tips and tricks - ArchWiki
  - Some good context for grub-mkimage.
- Syslinux - Gentoo Wiki
- Library modules - Syslinux Wiki
- RepackBootableISO - Debian Wiki
- Install - Syslinux Wiki
- File list of package syslinux-efi in stretch of architecture all
- sgdisk(8) - Linux man page
- Hybrid UEFI GPT + BIOS GPT/MBR boot
- "No suitable mode found" error
- Where is the memtest option on the Ubuntu 64-bit live CD?

- Remastering the Install ISO
- Install GRUB2 in (U)EFI systems
- 6.4 Embedding a configuration file into GRUB
- Standalone Grub2 EFI installation - grub.cfg placement?
- actual usage of 'grub-mkimage --config= '
- GRUB2 How To (4): memdisk and loopback device
- How does the grub efi loader find the correct grub.cfg and boot directory?
- How to Rescue a Non-booting GRUB 2 on Linux
- Using a CD
- Boot Linux with extlinux from EFI & GPT
- GRUB2 Modules
- grub-install.c
- Grub2/Troubleshooting
- Set up PreLoader
- Using PreLoader
- build.sh
- Archiso
- The simple menu system
- UEFI
- Make UEFI bootable live CD
- What is the proper way to use ISOLINUX with UEFI?
- Booting from removable media
- UEFI systems
- Managing EFI Boot Loaders for Linux: EFI Boot Loader Installation
- Examples
- Grub2 El-Torito CD
- Create ISO image with GRUB2
- DebianLive MultibootISO
- 11 GRUB image files
- stage2_eltorito missing
- 3.4 Making a GRUB bootable CD-ROM
- why is grub2 ignoring kernel options when boot from el torito on CD?
- Boot UEFI does not work from CD/DVD
- UEFI problem after upgrading to VMWS player 12
- make-efi

- UEFI + BIOS bootable live Debian stretch amd64 with persistence

*Feel free to contact me with questions or feedback regarding this article.*

subscribe via RSS

## Will Haley

Will Haley williamhaley