

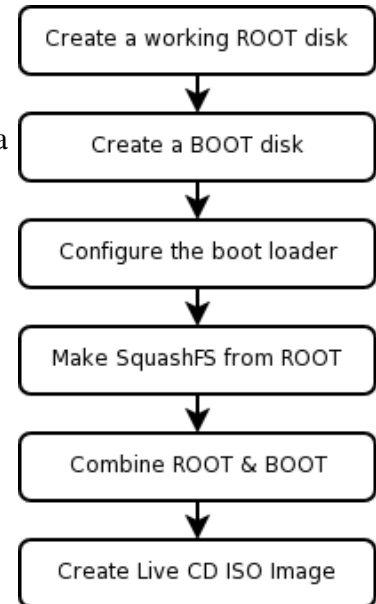
Building our Own Live CD: Step By Step

04/10/07 – First Draft
Copyright 2007 Jesse Caulfield
<jmc at kbg dot ch>

Abstract

Now that we know how to disassemble a Linux LiveCD, let's learn how to build our own. The process of creating a Linux Live-CD is essentially the same as process as decomposing one, only in reverse. We must first build a root disk for the system to use, then create a boot disk to load the system, and finally we combine them together with a boot loader to make our bootable CDROM. In summary:

- Create a working root disk
- Create a boot disk
- Configure the system boot loader
- Compress the root disk with squashfs
- Combine the compressed root and boot disk into a temporary staging area
- Create a Live-CD ISO from the combined boot and compressed root disks



Create a Root Disk

There are four principal methods for creating a Linux root disk; one is trivial, one is relatively easy, one somewhat involved, and the last is a labor of love. The method you choose is principally driven by your desired result and the family of software and tools you wish to include in your distribution.

1. Debian Family

Debian has a fully automated program for creating complete, functional new root disks called `debootstrap` (with a similarly functioning variant called `cdebootstrap`). `debootstrap` requires little to no configuration except specifications of what version should be built, where it should reside, and if there are any local mirrors to speed up the process.

A base system is approximately 90 MBytes and includes everything necessary to provide a system login. A large advantage to using the debian system is that the live-cd boot scripts needed in the initial ram disk are available as a package and require little to no modification.

2. Fedora Family

While no `debootstrap` equivalent exists, it is possible to populate a manually created and configured Fedora-based root disk using the `yum` program. It's a bit more work than `debootstrap`, and the base system comes in at around 90 Mbyte.

The principal disadvantage here is that the live-cd boot scripts must be configured and manually installed, meaning that the `initrd` image must also be manually created. There is one RPM package available to do all of this for you, but the live-cd scripts are manually created and not easily modified. The Fedora RPM is called `livecd-creator` written by David Zeuthen.

3. Busybox based

For small, embedded applications that do not require full unix system capability a busybox user-space is a good option.

Because busybox based systems have no concept of a run level, its common to use the initial ram disk as the actual root disk and create a streamlined boot process. Busybox booting and initialization is very simple but it can be an involved process to get programs like display managers working.

4. Manual build from Scratch

Lastly, it is possible to build an entire Linux distribution from source code. This task is not for those with short attention spans. More information about building from source code can be found at the Linux From Scratch web site.

5. Recap

In this document we'll focus on using debian based solutions. Debian has a very clever program to create a root disk called debootstrap, which requires a minimum of two parameters: the distribution type and its desired location. To create a debian root disk containing the latest software versions from the 'sid' suite, we need type only one command.

```
Usage: debootstrap [OPTION]... <suite> <target> [<mirror> [<script>]]
% debootstrap sid fs-root-debian-sid
```

Modify the Root disk

Once debootstrap is complete, you can enter your newly built system with the chroot command to modify it as necessary. Typical modifications may include adding additional packages, creating users and groups, and modifying start-up scripts.

```
% chroot fs-root
sid #
```

Once inside, you can install or remove software as you need with the built in Debian package management tools.

Create the Boot Disk

To create a boot disk we'll need three ingredients: a boot loader, a kernel, and an initial ram disk containing the proper boot scripts. These can all be copied from the host system or from the newly built root file system. If possible, my preference is to use contents from within the new root file system.

For a Live-CD boot loader, we can chose from three good options: lilo, grub and isolinux. My preference is grub. The initial ram disk is created later, and the

Create a boot working directory structure

```
# mkdir -p fs-boot/boot/grub
```

Mandatory packages for a Live-CD kernel include the 'unionfs' and 'squashfs' kernel modules.

Install a kernel and boot loader into the root file system

```
# chroot fs-root
```

```
sid # apt-get install kernel-image-xxx unionfs-modules-xxx squashfs-modules-xxx
```

where xxx is the kernel version you wish to install.

Download and install the boot loader files.

```
sid # apt-get install grub
```

Lastly, we must have a set of Live-CD initialization scripts. Debian contains a good working set in the 'casper' package, which require relatively little to no modification.

```
sid # apt-get install casper
```

Since we have installed new kernel modules and made changes to the initial boot scripts, we must rebuild the initial ram disk

```
sid # mkinitramfs
```

We can now copy the necessary files from the root file system and our boot file system is complete.

Copy the boot loader

```
# cp -a fs-root/usr/lib/grub/i386-pc/* fs-boot/boot/grub/
```

Copy the kernel and ram disk over to the boot disk

```
$ cp -a fs-root/boot/* fs-boot/boot/
```

Create the SquashFS Compressed Root Disk

Once our root disk has all the desired packages installed and is configured to our liking, we use the -all-root flash option of the 'mksquashfs' program to ensure the system permissions are correct

```
% mksquashfs fs-root fs-squashfs.squashfs -all-root
```

Combine ROOT & BOOT disks to Create the LiveCD ISO

Within the boot file system we must create a /casper directory, which is where the casper group of live-cd boot scripts we installed look for the root file system squashfs image. To create a bootable ISO image, copy and combine the contents of the root and boot file systems into the temporary working directory, then use the temporary directory as a source for the 'mkisofs' program.

For a Live-CD, the boot loader is identified at the time the ISO is created, in this case we grub's stage2_eltorito CDROM boot loader is specified.

```
% mkdir -p fs-temp/casper
% cp -a fs-boot/* fs-tmp
% cp fs-squashfs.squashfs fs-temp/casper/
% mkisofs -o Live-CD.iso -v -J -R -D -A Live-CD -V Live-CD \
-no-emul-boot -boot-info-table -boot-load-size 4 \
-b boot/grub/stage2_eltorito -c boot/boot.catalog fs-tmp
```

Note that the boot catalog is created by the mkisofs program and does not require our attention.

The resulting ISO image is called Live-CD.iso and can be burned to a CDROM or run in an emulator.

Appendix: Bootable USB Flash Drive vs. CDROM

A bootable flash drive is more similar to a bootable hard drive than a bootable CDROM, in that the root file system may exist on the the flash drive as an uncompressed, normal directory structure whereas on the CDROM it must be compressed and packaged into a SquashFS or CPIO loopback file system. A flash drive may optionally boot with a compressed root file system if desired.

To create a bootable flash drive, simply repeat the procedure to create a bootable hard drive or slightly

modify the procedure used to create a Live-CD, with the difference being that we must use a different boot loader.

For flash drives, we must install a stage-1 boot loader onto the flash drive and use the FAT stage-2 boot loader.

To start, insert and mount your flash drive, create the necessary directories, then copy over the boot and squashfs disk contents. Here we assume the USB flash drive is recognized as the device “/dev/sda1” and mounted in the directory “/media/FLASH”

```
% mkdir -p /media/FLASH/casper
% cp -a fs-boot/* /media/FLASH/
% cp fs-squashfs.squashfs /media/FLASH/casper/
```

Now we must install the stage 1 and 2 boot loaders to make the flash device bootable

```
% echo '(hd0) /dev/sda1' > /media/FLASH/boot/grub/devices.map
% grub-install --root-directory=/media/FLASH --nofloppy '(hd0)'
```

You should now have a bootable flash drive. Enjoy!