Install Slackware using a bootable USB stick



Please note that starting with Slackware 12.0 installing from USB stick or from the network (PXE boot) is supported out of the box!
You can find the USB bootable image file called `usbboot.img` in the **/usb-and-pxe-installers** directory. Also, Slackware contains modified versions of my Wiki pages in that same directory. They are called **README_USB.TXT** and **README_PXE.TXT**.
The remainder of this Wiki article is basically preserved here as a technical reference, but you are no longer required to follow all the instructions below. Check out the Slackware *README*s instead.

Introduction



Recently I have been exploring the Slackware installer, and especially looking for ways to use the installer other than running it from a CDROM/DVD. In a previous wiki article I detailed my experiences with creating a PXE installer that you can use if your computer is able to do a *network boot*. This is also fondly called PXE boot. This was fun, and it was the end of using CDROMs for my Slackware installations - as long as I had a PXE server configured in the local network. Of course, when you visit other places, such a server is not always available



So, I went on, and turned to USB sticks, also known as *pen drives* or *thumb drives*. These devices are small,lightweight, easy to carry around and modern PC's are usually able to boot from a suitable USB medium (such as the pendrive).

The result of my experiments is documented in this Wiki page. I will show you two methods of using a bootable USB stick to install Slackware. The first method - creating the USB equivalent of a boot/root floppy pair - is relatively easy to implement. The second method took quite a bit more research. It implements an image file containing two partitions. One (with a fat filesystem) contains the boot kernels and the root filesystem (initrd), while the other partition (fitted with a ext2 filesystem) contains packages to install. So, while the first method still needs a NFS server or another source of packages (such as a CDROM or a local hard disk partition), the second method is self-contained. If the USB stick is large enough (1 to 2 GB) you can carry the full Slackware install with you on your key chain! But even a small (256 MB) USB stick has room for the “a”, “ap” and “n” package series, so you can install a functional network-capable and console-based Slackware environment off it.

Note:
I am taking the convoluted path of first creating an image file and then copying that to a USB stick. It is easier to directly write to a USB stick and forget about the intermediate stage of creating the image file, and particularly much easier in the case of the multi-partition USB stick...
But USB Flash devices are not meant to be written an infinite amount of times, they have a limited lifetime regarding write operations. I wanted to keep my USB sticks out of harm's way, and also I should mention that writing to a file on hard disk is much faster than writing to the USB device. This sped up the development process considerably. Finally, it was a great learning experience in working with loop devices and image files.

Boot disk on a mini USB image

I had no idea how bootable USB sticks were configured, but by looking at the Slackware CD's *isolinux* bootloader, it turned out that it's sibling *syslinux* was the ideal candidate for creation of such a bootable USB image. I started small, by creating an image containing a single partition to which I copied the contents of the Slackware boot disk (i.e. large parts of the `/isolinux` directory of the Slackware package tree and then some more). This worked great!

The result is a 37 MB small USB boot image that you can use to boot into the Slackware setup program (and then use a NFS server or another Slackware package source like a prepared local harddisk partition).The small image works great, especially when you don't have or don't want to use CDROM media as the carrier for the Slackware packages. I myself install Slackware from my own NFS server because my network is blazing fast compared to the much slower CDROM data transfer speeds.
The USB image is small enough to make it available for download on a web server, and is transfered to a USB stick in a matter of seconds. Even the oldest and tiniest of USB drives is well suited for this purpose.

So, how does one create this small image file, and how is it used?

We will need access to a local mirror of the Slackware directory tree, or at least the **`isolinux`** and **`kernels`** directories thereof. In the examples below, I am assuming the Slackware tree is located at **`/mirror/Slackware/slackware-11.0`**. You will have to adapt the example commands so that they are correct for your own situation.

The mini image will not contain any installable Slackware package. Instead, we are relying on a nearby NFS server from which we can retrieve the packages. The usual way of installing Slackware from a NFS server repository, is to boot the Slackware CDROM, and run (`pcmcia` and) `network` to load all the drivers for your network card before starting `setup` to commence the installation procedure. When you type `pcmcia` and/or `network`, the Slackware installer will look for the appropriate driver files on an available CDROM, and if no CDROM is found it prompts you to insert floppy disks.
Now, with a boot from our USB stick we are assuming there is *no CDROM* and *no floppy* drive available to us. So, we need to add all the drivers we need to the root filesystem where the setup program and related stuff is stored (this is what is in the `initrd.img` file - *initrd* stands for *initial ramdisk*).

If you want the easy way out, you can either download the ready-made image file or download the script I wrote and which implements all the steps I'm about to describe. Otherwise, read on.

Creating a new initrd.img file

Let us do our work in a temporary directory I call the *staging* directory:

```
mkdir -p /tmp/slackboot
```

The fun thing is, a lot of the work looks exactly like the steps I described in my PXE installer article. Instead of duplicating my efforts, I am sending you over to that section of the article to read exactly how you build a modified `initrd.img`.

Assembling the USB image

Now that we have a `initrd.img` file with all the functionality we need, it is time to create the larger USB image file.

- First, we calculate the sizes of the files we are going to copy into the USB image (the `initrd.img` and the boot kernels are the bulk). That target file must be larger than the sum of the sizes of those source files:

```
let USBIMG=$( du -sk /tmp/slackboot/initrd.img | cut -f1 )
for KERN in /mirror/Slackware/slackware-11.0/kernels/*.?/*zImage ; do
  let USBIMG=USBIMG+$(du -sk $KERN | cut -f1 )
done
let USBIMG=USBIMG+512  # Add just that little extra..
```

  This leaves us with a variable `${USBIMG}` whose value is the estimated size in kB of the file we are about to create. You could of course also calculate this value by using your eyes and brain and looking at the `ls -l` output  But the exercise is meant to enable you to copy/paste the example commands, and to create your own script from it.

- Create a DOS formatted image file:

```
mkfs.msdos -n USBSLACK -F 16 -C /tmp/slackboot/usbboot.img ${USBIMG}
```

  You will notice that we use the switch "`-C`" and the variable `${USBIMG}` that we calculated in the previous step. With the `mkfs.msdos` commandline like that, we can omit the step where we would use the "`dd`" command to create the file prior to formatting it - `mkfs.msdos` will do that for us at no cost.
  The command also labels the new FAT filesystem with the name *USBSLACK* (the "`-n` USBSLACK" option) but this is not required.
  The important piece of information that is contained in the command is the fact that this FAT filesystem **must** be a FAT16 filesystem (`-F 16`). It is tempting to use FAT32 but the `syslinux` bootloader will fail to make the USB stick bootable if the filesystem is not FAT16.

  The file `/tmp/slackboot/usbboot.img` will eventually be the file to copy to the USB stick.

- Loop-mount the image so that we have a mount point to where we can start copying files:

```
mkdir -p /tmp/slackboot/usb
mount -o loop,rw /tmp/slackboot/usbboot.img /tmp/slackboot/usb
```

- Now, we copy the files that syslinux needs:

```
cp /mirror/Slackware/slackware-11.0/isolinux/isolinux.bin
/tmp/slackboot/usb
cp /mirror/Slackware/slackware-11.0/isolinux/setpkg /tmp/slackboot/usb/
cp /mirror/Slackware/slackware-11.0/isolinux/{f*.txt,message.txt}
/tmp/slackboot/usb/
```

```
cp /tmp/slackboot/initrd.img /tmp/slackboot/usb/
cat /tmp/slackboot/pxelinux.cfg_default | sed -e 's# /kernels/# #g' -e
's#/.zImage##' > /tmp/slackboot/usb/syslinux.cfg
```

Note:
The file `/tmp/slackboot/pxelinux.cfg_default` was created together with our modified `initrd.img` file. The difference with the file `/mirror/Slackware/slackware-11.0/isolinux/isolinux.cfg` which you perhaps expected me to use, it that we need a much bigger ramdisk than the Slackware bootable CDROM, because we added all the network and pcmcia stuff to the initial ramdisk.
This of course means that this is not a suitable method to boot old PC's that are low on memory! The ramdisk will be more than 16 MB in size, and you will need RAM for your kernel as well.

- Syslinux can not cope with subdirectories, that is the reason we need to place all the files that syslinux uses in the root of the loopmounted FAT filesystem - including the bootkernels:

```
cd /mirror/Slackware/slackware-11.0/kernels/
for dir in `find  -type d -name "*.?" -maxdepth 1` ; do cp $dir/*zImage
/tmp/slackboot/usb/$dir ; done
```

- Finally, we “stamp” the file with the syslinux bootloader:

```
umount /tmp/slackboot/usb
syslinux -s /tmp/slackboot/usbboot.img
```

This creates the file “`ldlinux.sys`” in the root of the partition; it contains the bootloader code. If you ever change the content of the `syslinux.cfg` file, you will have to run the command `syslinux -s /tmp/slackboot/usbboot.img` again, to update the bootloader.

Note:
Open the file `~/.mtoolsrc` in an editor and add the line “mtools_skip_check=1” in case the `syslinux` command gives an error.

- This concludes the steps needed to create our image file! All that's left is to copy the file to a USB stick:

```
dd if=/tmp/slackboot/usbboot.img of=/dev/sda bs=512
```



Be careful about the device name for your USB stick! The above dd command will wipe out any existing data on the device, so you had better be sure that it is not the SATA hard disk you're targeting!

Booting from the stick



your computer BIOS must support booting from “USB HDD”

Plug the stick into your computer's USB slot, and boot it up. Make sure you select “*boot from USB-HDD*” - how you should do this depends on the type of computer you have.
Many computers will display a message during the initial stages of the booting that says something like “`Press [F12] for a boot device list`”.

The Slackware installer will start just like when you had booted from a CDROM. Log in as “root”. Start the install by partitioning your hard drive as usual, and running `setup`.
If you want to install from a NFS server, you should run the commands `pcmcia` (if your network card is PCMCIA) and/or `network` prior to running `setup` in order to load a driver for your network card.

If you want to install Slackware using a local hard disk partition in case you copied the Slackware tree there in advance, that is also an option. It would not make much sense to opt for the third install method “use a CDROM” (or DVD) since we just abandoned the use of a CDROM medium 🙂
On the other hand, I once dealt with an IBM server that was not able to boot off a DVD containing a distro that shall remain unnamed, but was able to boot from a USB stick. Remember, Slackware 11.0 is available on DVD, too! Many older servers are not equipped with a DVD drive, so a bootable USB stick will save the day.



After writing the usbboot.img to the USB stick, if you run `fdisk -l` you will see alarming output like this:

```
This doesn't look like a partition table
Probably you selected the wrong device.

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   ?     8563200     8326647  2088818490    1  FAT12
 Partition 1 has different physical/logical beginnings (non-
Linux?):
     phys=(124, 38, 11) logical=(8563199, 1, 16)
 Partition 1 has different physical/logical endings:
     phys=(344, 195, 26) logical=(8326646, 0, 49)
 Partition 1 does not end on cylinder boundary.
```

... and so on, for partitions 2, 3 and 4 as well.
This is actually harmless. The usbboot.img uses the “raw” device, it did not create partitions at all. Fdisk reads the information in the first sector and incorrectly interprets that as a screwed-up partition table. Just ignore the warnings.

Evaluation

When I finally had my USB bootable pendrive, I sat back and evaluated.
What I had was nice enough, and a desirable addition, but I wanted to take this concept of a bootable Slackware USB stick a little further even. What if it were possible to copy all (or most) of the

Slackware CD's to a bootable USB stick? You could carry your Slackware with you all the time and install it on your Mom's computer on your next visit (but she probably won't have the PC that is able to boot from a USB stick anyways...).

A complete Slackware setup on USB image

In the remainder of this article I want to explain the steps you have to take (aka the hoops you have to go through) in order to create this USB image file. No pendrives are being hurt in the development process - we are going to do this on our hard disk where we have access to a Slackware tree (you know, the directory structure with the `/isolinux`, `/kernels`, `/slackware`, `/extra` and many more directories at it's root).
We will end up with a single file that we can transfer to a USB stick.

It all depends on the size of your USB stick whether you can copy all packages from the Slackware CDROMs onto it (so you can install KDE and a 2.6 kernel) or add only the basic packages to the image (take at least the “a” “ap” and “n” package sets). To contain all the packages that make a full Slackware install you'll need a USB stick that is larger than 1GB ...

The reason for needing two partitions is that our bootloader *syslinux* requires the filesystem on which it is installed to be **FAT16**. This limits the filesystem to 2 GB and I wanted to be able to use the new 4 GB USB pen drives that are on the market. So, I kept the FAT partition to a minimum and crammed all the packages and other stuff into a ext2 partition.



This part remains to be written


Restoring a USB stick to its original state

When you have used the small 40 MB image to create a USB installer, your USB stick is no longer useful for anything else. Any remaining space on the stick (supposing you used a larger-than 40 MB stick for it) is inaccessible.
Fortunately, it is easy to re-create a FAT partition on the stick (thereby removing the Slackware installer of course) so that the USB stick again becomes available for carrying around your data.


Take care about which device actually is your USB stick !!! The next command will render all present data on `/dev/sdX` inaccessible by deleting it's partition table!!!

- First, wipe the bootsector of the USB stick:

```
dd if=/dev/zero of=/dev/sdX bs=512 count=1
```

- Then, create a new FAT16 partition (*type* '*6*' in fdisk terminology) on the stick and write a FAT32 (vfat) filesystem on it:

```
fdisk /dev/sdX <<EOF
n
```

```
p
1



t
6
w
EOF
mkdosfs -F32 /dev/sdX1
```

The 10 lines starting with "`fdisk /dev/sdX «EOF`" and ending with the single word "EOF" are actually one single command spread over ten lines, *including* the two empty lines. This format is called a here-document. It allows us to use a command which expects interactive input (fdisk) non-interactively, in a shell script for instance. If you're uncomfortable with it you can just run

```
fdisk /dev/sdX
```

and create a partition interactively 🙂

From:
https://wiki.alienbase.nl/ - **Alien's Wiki**

Permanent link:
https://wiki.alienbase.nl/doku.php?id=slackware:usbboot

Last update: **2009/11/27 13:20**