Move an installed Linux distribution, with its configuration and user space, from HDD to a USB drive. Make the USB drive bootable in either legacy mode or UEFI mode

Neville Jackson

2 Apr 2022

1 Introduction

Sometimes one needs to save an installed Linux, but keep it available for occasional viewing. A conventional backup of its partition(s) will do , but it is inconvenient to have to restore it to use it. A convenient solution is to copy it to a USB drive, and make the drive bootable.

There are two ways of transfering a running Linux to a USB drive. There is a good FOSS article [1] on this issue. One is to make an .iso file and image that to the USB drive and make it bootable. This results in what is commonly called a *live filesystem*. It can be booted and used , but any changes are lost on shutdown. There are ways of adding *permanence* to a live filesystem, but they are complicated and limited, and do not allow kernel updates.

The second way is to do the equivalent of a full linux install on the USB drive. This allows it to operate just like a hard disk install with full permanence and updateability. This document looks at implementing this second option.

2 Steps to implement full Linux transfer to USB drive

We need to prepare partitions on the USB flash drive, transfer a copy of the required Linux from HDD, mount the new copy and fix the file /etc/fstab, install grub on the USB drive, boot grub to the *grub* command menu, boot the Linux copy on the USB drive from grub comand line, then use the booted Linux to configure grub.

2.1 Prepare partitions

Use an installed Linux that has *gparted* or a gparted DVD or USB drive. Wipe the UBB drive clean. Make at least 4 partitions

1. EFI_System partition, fat32, 512 Mb, boot,esp flags
2. BIOS-boot partition, no fileasystem, 1Mb bios_grub flag
3. LinuxRoot partition, ext4 , at least 20Gb, mount point /
4. Linux_swap partition, swap, 4GB
5. others as needed

The EFI_System partition should be first.

2.2 Copy Linux root partition from HDD to USB disk

There are various ways of copying and entire Linux root filesystem. I used *rsync* but *dd* is an option. There is no need to image the root filesystem, just copy it.

1. Boot any Linux, preferably not the one to be copied
2. Mount the Linux filesystem to be copied – in my case the mount is
   mount /dev/sdb12 /media/nevj/Linuxroothome
3. Check the name of the USB drive partition to be copied to
   lsblk will list all disk partitions, mounted or not. In my case it is /dev/sdc3
4. mount /dev/sdc3 /mnt
5. rsync -aAXvH –exclude={'dev/*','proc/*','sys/*','tmp/*','run/*','mnt/*'
   ,'media/*','lost+found','common/*'} /media/nevj/Linuxroothome/ /mnt
   This will copy all directories except those excluded.
   In my case common is a data partition, I want to exclude that.
6. do a *sync* before proceeding

The –*exclude* option on *rsync* avoids copying psuedo filesystems that are populated at boot time and any mounts, especially /mnt which is the USB drive partition copied to.

2.2.1 Note on rsync

The –*exclude*{*'dev/*',....*} option on *rsync* used above used *brace expansion*. That only works if your shell is *bash* or *csh*.

You can test if it is going to work by doing

```
$ echo {a,b,c}
a b c
```

if it does not work you will get

```
$ echo {a,b,c}
{a,b,c}
```

If you get the latter, start *bash* before you run the *rsync* command. If you do not have *bash* download and install it. As a last resort you can modify the *rsync* command as follows.

```
rsync -aAXvH --exclude='dev/*' --exclude='proc/*' --exclude='sys/*'
  --exclude='tmp/*' --exclude='run/*' --exclude='mnt/*'
   --exclude='media/*' --exclude='lost+found' --exclude='common/*'
   /media/nevj/Linuxroothome/ /mnt
```

That should work in any shell. It does not matter whether the quotes are single or double.

2.2.2 Copying from within an active Linux filesystem

This can be done. One should halt all user processes that are likely to write files first. The *rsync* command is slightly different in this case

```
rsync -aAXvH --exclude=\{'dev/*','proc/*','sys/*','tmp/*','run/*','mnt/*'
      ,'media/*','lost+found','common/*'\} / /mnt
```

because one does not need to mount the root directory

2.3 Patch the /etc/fstab file and remove grub configuration

There may be entries in */etc/fstab* which will need to be changed. In particular UUID's will need to be set to the correct values for any USB drive partitions.

1. Find the UUID numbers of the partitions on the USB drive. Use a disk utility or
   ls -l /dev/disk/by-uuid
2. Edit /etc/fstab on the root partition on the USB drive. . Carefully copy the UUID's into fstab. You will need at least one for the root filesystem, and one for the swap partition.

The resultant /etc/fstab should look as follows

```
# Pluggable devices are handled by uDev, they are not in fstab
# / sdc3
UUID=0beb5819-f2ba-4fa0-aa69-3e5ec16fb0bc  /   ext4   noatime   1 1
# swap sdc5
UUID=ef94e2d5-c924-49b8-a944-486efd629340  swap  swap   noatime   1 2

# spare partition sdc4
```

```
UUID=d4109c75-0428-4bb9-8d19-d0b63d09930a /home/nevj/spare ext4 0,users 2 0

# common partition - filesystem shared by several os's
#/dev/sda4    /common      ext4      rw                0       2
```

The only essentials are / and swap. I have an extra partition called spare, and I have commented out an HD partition called /common. There should be no HD partitions because the USB drive willl need to work in a self contained manner on any computer. I am not sure whether the / entry is needed.

If the Linux copied to USB drive has had grub configured in the HD copy, it will be necesssary to remove the grub configuration in the USB copy. Mine did not have this. Just go to /mnt/boot (ie on the USB drive) and do

```
rm -r grub
rm -r efi
```

2.4 Install grub on USB drive

Consult the GNU Grub manual [2]. Login to any Linux on the HD. Check the partition names of the USB drive using *lsblk*. Mount the root directory of the USB drive as follows

```
mount /dev/sdc3 /mnt
```

where sdc3 is the root directory. Then install grub as follows

```
grub-install --boot-directory=/mnt/boot
             --recheck
             --removable
             --targer=i386-pc
             /dev/sdc
```

where sdc is the usb drive device name. This installs grub for a *legacy mode* boot.

2.5 Use grub command line to boot Linux on USB drive

Reboot the computer and use the BIOS to boot from the USB drive. You should get the grub command prompt

```
grub>
```

If you get anything else there is an error. There is no grub menue, because we have not configured grub yet on the USB disk.

We can now use grub commands to boot the copy of Linux which is on the USB disk as follows

```
grub> linux /vmlinuz root=/dev/sdc3
grub> initrd /initrd.img
grub> boot
```

and it should boot. Login

2.6 Use the booted USB drive Linux to configure grub on the USB drive

Now that we have the USB drive copy of Linux booted, we can use it to do its own grub configuration. That is easy

```
Edit /etc/default/grub, adding or modifying the line
GRUB_DISABLE_OS_PROBER=true
then simply
update-grub
```

The update-grub should find the Linux on the USB drive, but not find any other Linuxes on the HDD. That is what we want - we want the USB drive and its grub to be configured independently of the harddisk.

While there you can test that the swap space is mounted

```
swapon --show
```

Having its own swap space is part of making the USB drive independent

Test any other mounts, as required.

2.7 Test boot

Reboot the computer and use BIOS agian to boot the USB drive. This time it should bring up a grub menu instead of a command line with just 3 entries - your copied Linux, your copied Linux again in Advanced mode, and maybe Memtest if it comes with your grub. Check out booting from the menu.

Then the acid test. Shutdown, remove the USB drive, put it in another computer, and boot it there. Mine worked, I hope yours does too.

2.8 Booting in UEFI mode

There seems no reason why one can not do another grub-install to the same USB drive only using the EFI-System parttion instead of the BIOS-grub partition. The drive would then be bootable in either mode. The two grubs would share a configuration, so one should not have to repeat the update-grub step.

So lets do it. Mount the root directory of the USB drive as follows

```
mount /dev/sdc3 /mnt
```

where sdc3 is the root directory (use *lsblk* to check, then install grub as follows

```
grub-install  --boot-directory=/mnt/boot
              --recheck
              --removable
              --target=x86_64-efi
              --efi-directory=/dev/sdc1
```

Notice that we dont specify a device , because the efi grub install does not write on the device, it writes in the EFI-System partiton. This installs grub in UEFI mode.

We dont need to repeat Section 2.5 or 2.6, because the grub configuration on the USB drive is already done using legacy grub. The two grubs share the same configuration files on te USB drive.

2.9 Test boot in UEFI mode

Reboot and set the BIOS to UEFI mode and choose the USB drive. The grub menu belonging to the USB drive wil appear. Check out booting from this menu. Then the acid test - boot the USB drive in another computer in UEFI mode. Mine works. I wish you luck.

2.10 Acknowledgment

Feedback from Edgar Hoffman is appreciated.

References

[1] Emmanuel (2021) Persistent Live USB vs. Full Linux Install on a USB Drive. URL https://www.fosslinux.com/49280/persistent-live-usb-vs-full-linux-install-usb-drive.htm

[2] Matzigjkeit, Okuji, Watson, and Bennett (2021) The GNU GRUB Manual URL https://www.gnu.org/software/grub/manual/grub/grub.html