How to configure fstab and lilo.conf with persistent naming

Have you ever had your /dev/sda and /dev/sdb drives switch order? Has plugging in a USB device before booting your computer caused a kernel panic? Do you frequently unplug or plug in new devices into your computer, thus changing the order of the device names? If you said yes to any of those, you might be a good candidate to use persistent naming within your bootloader and fstab.

What is persistent naming?

Persistent naming allows you to reference your drive by something that doesn't change. There are several different methods available for persistent naming and we'll be covering the following: ID, Label, and UUID, as well as the GPT-based PartLabel and PartUUID. The most frequently used are labels and UUIDs, mainly because they have been around for a long time. GPT partition tables introduced an updated version of labels and UUIDs called partlabel and partuuid that are directly stored in the partition table. However, starting with the 3.8 kernel, PartUUID has been expanded to generate shorter unique identifiers (UID) for MBR partitions as well, essentially allowing MBR drives to use PARTUUID.

Now, let's go more in depth on the various methods:

ID

`/dev/disk/by-id/` is a dynamically generated folder under the `/dev/` filesystem. This will change depending on the devices that are plugged into the computer. The IDs are generally comprised of the harddrive's model number, serial number, and specific partitions. `ls -la /dev/disk/by-id/` will show all the IDs and the device it points to under the `/dev/` filesystem (e.g. `/dev/sda`, `/dev/sdc3`, etc).

```
slackware@darkstar:~$ ls -la /dev/disk/by-*
/dev/disk/by-id:
total 0
drwxr-xr-x 2 root root 680 Jul 31 13:17 ./
drwxr-xr-x 8 root root 160 Jul 31 13:17 ../
lrwxrwxrwx 1 root root 9 Jul 31 17:18 ata-Crucial_CT480M500SSD1_13440956798B
-> ../../sda
lrwxrwxrwx 1 root root 10 Jul 31 17:18 ata-
Crucial_CT480M500SSD1_13440956798B-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Jul 31 17:18 ata-
Crucial_CT480M500SSD1_13440956798B-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 Jul 31 17:18 ata-
Crucial_CT480M500SSD1_13440956798B-part3 -> ../../sda3
lrwxrwxrwx 1 root root 9 Jul 31 17:18 ata-WDC_WD1001FALS-00J7B1_WD-
WMATV1953756 -> ../../sdb
lrwxrwxrwx 1 root root 10 Jul 31 17:18 ata-WDC_WD1001FALS-00J7B1_WD-
```

```
WMATV1953756-part1 -> ../../sdb1
lrwxrwxrwx 1 root root 9 Jul 31 17:18 ata-WDC_WD40EZRX-00SPEB0_WD-
WCC4E1021582 -> ../../sdc
lrwxrwxrwx 1 root root 10 Jul 31 17:18 ata-WDC_WD40EZRX-00SPEB0_WD-
WCC4E1021582-part1 -> ../../sdc1
```

So, if you want to always reference the Crucial drive (just the drive, not any partitions within it it), you can use the filename `/dev/disk/by-id/ata-Crucial_CT480M500SSD1_13440956798B`. (You will see an example of this in the lilo.conf section below.) This will always point to that drive, regardless of whether it is `/dev/sda` or `/dev/sdq`. To reference a specific partition, just match the current device name (`/dev/sdb1`) with its associated ID, `/dev/disk/by-id/ata-WDC_WD40EZRX-00SPEB0_WD-WCC4E1050616-part1`.

Label

Labels are an optional, user configured name for the drive. Most filesystems support this, but they aren't always created automatically. All labels can be found under `/dev/disk/by-label/` and can be easily viewed using `ls -la /dev/disk/by-label/`.

```
/dev/disk/by-label:
total 0
drwxr-xr-x 2 root root 80 Jul 31 13:17 ./
drwxr-xr-x 8 root root 160 Jul 31 13:17 ../
lrwxrwxrwx 1 root root 10 Jul 31 17:18 Backup -> ../../sdc1
lrwxrwxrwx 1 root root 10 Jul 31 17:18 Data -> ../../sdb1
```

Labels can be quite advantageous over IDs due to their ability to be extremely short and memorable. However, if you're not careful or move disks between systems, you can easily run into issues with duplicate names causing conflicts. Both lilo and fstab support using LABEL=“Backup” to reference the drive directly rather than needing to use `/dev/disk/by-label/Backup`, making this a much simpler option to reference drives.

If you want to add a label to your partition, the ArchWiki page has instructions for most filesystems.

UUID

UUIDs (**U**niversally **U**nique **ID**entifiers) are a unique identifier generated for a partition. All Linux filesystems will generate UUIDs when they're formatted. FAT and NTFS filesystems do not support UUIDs, however, a short UID is used in place of a regular UUID. All UUIDs can be found under `/dev/disk/by-uuid/` and can be viewed using `ls -la /dev/disk/by-uuid/`

```
/dev/disk/by-uuid:
total 0
drwxr-xr-x 2 root root 200 Jul 31 13:17 ./
drwxr-xr-x 8 root root 160 Jul 31 13:17 ../
lrwxrwxrwx 1 root root 10 Jul 31 17:18 226f1db3-2f14-4862-9998-8be1e8f7cdc9
-> ../../sda3
```

```
lrwxrwxrwx 1 root root 10 Jul 31 17:18 23bce2c2-996d-449e-89cc-0e5029cc6d8d
-> ../../sda2
lrwxrwxrwx 1 root root 10 Jul 31 17:18 bc4538e5-9387-4e5c-877b-d15744acb6ce
-> ../../sda1
lrwxrwxrwx 1 root root 10 Jul 31 17:18 cd554cf8-c346-4bd5-ba6a-84ac99af571f
-> ../../sdc1
lrwxrwxrwx 1 root root 10 Jul 31 17:18 2ef6e776-c2eb-4e56-bc7e-dfe083559d8f
-> ../../sdb1
lrwxrwxrwx 1 root root 10 Jul 31 17:18 126AC3506AC32EF3 -> ../../sdq1
#Example NTFS UID -- not really in my system
```

UUIDs can also be referenced by using just `UUID=226f1db3-2f14-4862-9998-8be1e8f7cdc9` in your `/etc/lilo.conf` and `/etc/fstab` files.

GPT (PartLabel and PartUUID)

The MBR (Master Boot Record) has been the primary partition table for harddrives for the past few decades (it was introduced in 1983). However, it is unable to be used on disks larger than 2TBs. That is where GPT (GUID Partition Table) comes in. This was developed as part of the UEFI spec but can be used without using UEFI.

In addition to the limitations of the MBR, there are also limitations of using UUIDs and Labels, mainly being that if you use those in your bootloader, you are required to use an initrd. While it is recommended by Pat to use an initrd, many users prefer using a kernel that doesn't require one (either using the huge kernel or building a custom kernel with the required hardware support compiled in). The GPT spec added PartLabel and PARTUUID as a way to save that information directly to the partition table (the “part” stands for partition). While both PartLabel and PartUUID are included in the GPT spec, the kernel only supports booting PARTUUIDs, so you can't use PartLabel in your bootloaders. If you want to use PartUUID, you don't need any initrd to be able to read this data, which makes it much easier to use on systems that aren't already using an initrd. Initially, since PARTUUID was introduced as part of the GPT spec, MBR-based devices could not be referenced using PARTUUID; however, basic support for PartUUID on MBR partitions was added starting with the 3.8 Linux kernel, so now any drive can be referenced by its “PARTUUID”.

PartLabel

This is almost identical to the regular Label option. It is located under `/dev/disk/by-partlabel/` and can be viewed with `ls -la /dev/disk/by-partlabel`. PARTLABEL can only be used in the fstab. Due to lack of support in the kernel, you will need to use one of the other persistent naming conventions in your lilo.conf

```
/dev/disk/by-partlabel:
total 0
drwxr-xr-x 2 root root 60 Jul 31 17:18 ./
drwxr-xr-x 8 root root 160 Jul 31 13:17 ../
lrwxrwxrwx 1 root root 10 Jul 31 17:18 Linux\\x20filesystem -> ../../sdc1
```

Also, this will only show official PARTLABELs. It will not show any labels from disks using MBR. If you

need to view those, you can use `blkid`.

```
slackware@darkstar:~$ su -
Password:
root@darkstar:~# blkid
/dev/sdc1: LABEL="Backup" UUID="cd554cf8-c346-4bd5-ba6a-84ac99af571f"
TYPE="ext4" PARTLABEL="Linux filesystem" PARTUUID="62c372af-e868-4571-8ab4-
db612c0fb38f"
/dev/sdb1: LABEL="Data" UUID="2ef6e776-c2eb-4e56-bc7e-dfe083559d8f"
TYPE="ext4" PARTUUID="71b70c68-01"
/dev/sda1: UUID="bc4538e5-9387-4e5c-877b-d15744acb6ce" TYPE="swap"
PARTUUID="6f47c81b-01"
/dev/sda2: UUID="23bce2c2-996d-449e-89cc-0e5029cc6d8d" TYPE="ext4"
PARTUUID="6f47c81b-02"
/dev/sda3: UUID="226f1db3-2f14-4862-9998-8be1e8f7cdc9" TYPE="ext4"
PARTUUID="6f47c81b-03"
```

As you can see in the case of `/dev/sdc1`, you can have a different LABEL and PARTLABEL. That is because these are stored independently of one another. LABEL is a filesystem trait, whereas PARTLABEL is a partition trait. Also note, based on how I created my filesystem, it has a generic “Linux filesystem” PARTLABEL, so if I were to create a new filesystem on a new drive without changing anything, I would have two drives with the same PARTLABEL. This is not an issue unless you try to reference the disk as a PARTLABEL.

If you want to add or change your PARTLABEL, you can do so using `gdisk` or `cgdisk`.

PartUUID

This is similar to PartLabel in that it is almost identical to it's not-“Part” counterpart (regular UUID). This UUID is generated automatically whenever you create/resize/move a partition. This will be a different UUID than the one provided by the filesystem. All PartUUIDs can be found under `/dev/disk/by-partuuid/` and can be viewed using `ls -la /dev/disk/by-partuuid/`.

```
/dev/disk/by-partuuid:
total 0
drwxr-xr-x 2 root root 120 Jul 31 13:17 ./
drwxr-xr-x 8 root root 160 Jul 31 13:17 ../
lrwxrwxrwx 1 root root 10 Jul 31 17:18 468a6e66-39e3-4072-8997-0550dd12ad8e
-> ../../sdb1
```

As you can see, this is just like PartLabel in that it only shows official PartUUIDs on GPT disks. If you need to view the system generated PartUUID for MBR partitions, you need to use blkid.

```
slackware@darkstar:~$ su -
Password:
root@darkstar:~# blkid
/dev/sdc1: LABEL="Backup" UUID="cd554cf8-c346-4bd5-ba6a-84ac99af571f"
TYPE="ext4" PARTLABEL="Linux filesystem" PARTUUID="62c372af-e868-4571-8ab4-
```

```
db612c0fb38f"
/dev/sdb1: LABEL="Data" UUID="2ef6e776-c2eb-4e56-bc7e-dfe083559d8f"
TYPE="ext4" PARTUUID="71b70c68-01"
/dev/sda1: UUID="bc4538e5-9387-4e5c-877b-d15744acb6ce" TYPE="swap"
PARTUUID="6f47c81b-01"
/dev/sda2: UUID="23bce2c2-996d-449e-89cc-0e5029cc6d8d" TYPE="ext4"
PARTUUID="6f47c81b-02"
/dev/sda3: UUID="226f1db3-2f14-4862-9998-8be1e8f7cdc9" TYPE="ext4"
PARTUUID="6f47c81b-03"
```

For non-GPT filesystems, the PARTUUID is generated from the Disk Identifier, followed by the partition number. You can view your Disk Identifier using fdisk.

```
root@darkstar:~# fdisk -l /dev/sda | grep identifier
Disk identifier: 0x6f47c81b
```

Due to the newness of GPT support, lilo does not have direct support of calling it, rather you need to use an addappend option (see the lilo.conf section below for further information). For PartUUIDs on MBR disks, the only time they'll change is if additional partitions are created. Keep that in mind if you're using them in your fstab and/or lilo.conf.

fstab

Your `/etc/fstab` is the easiest file to change. You simply replace any entries that pointed to your device names with TYPE=value (TYPE can be LABEL, UUID, PARTLABEL, PARTUUID). The original device names are listed as comments for your reference, they don't need to be in the final file. (Personally, since I like it to look nice and neat, I will make everything spaced out evenly.)

```
slackware@darkstar:~$ cat /etc/fstab
# /dev/sda1
UUID=bc4538e5-9387-4e5c-877b-d15744acb6ce      swap            swap
defaults,discard                    0 0
# /dev/sda2
UUID=23bce2c2-996d-449e-89cc-0e5029cc6d8d      /               ext4
defaults,noatime,discard            0 1
# /dev/sda3
PARTUUID="6f47c81b-03"                         /home           ext4
defaults,noatime,discard            0 2
# /dev/sdb1
LABEL=Data                                     /share/Data     ext4
defaults                            0 2
# /dev/sdc1
PARTLABEL="Linux filesystem"                   /share/Backup   ext4
defaults                            0 2

#/dev/cdrom                                    /mnt/cdrom      auto
```

```
noauto,owner,ro,comment=x-gvfs-show  0 0
devpts                                      /dev/pts        devpts
gid=5,mode=620                       0 0
proc                                        /proc           proc
defaults                             0 0
tmpfs                                       /dev/shm        tmpfs
defaults                             0 0
```

lilo.conf

Your `/etc/lilo.conf` takes a bit more finesse to change. Not everything is properly supported like the fstab options, so we need to throw in a few workarounds, depending on what you want to use.

To start out, we need to set up the top of your `/etc/lilo.conf`. This mainly includes the boot entry. This is what lilo uses to determine where to write the bootloader code into the partition table. This is only referenced when you run lilo to install the bootloader, so it won't be as affected as other drives if they're moved. However, if you've already taken the time to get this far, you might as well ensure you don't run into any issues by referencing your drive using persistent naming.

To do this, we would need to replace `/dev/sda` (or whatever your default boot drive is) with a reference that doesn't change. Since we need to reference the drive itself, and not any specific partitions, we need to use the ID method. Once you find the disk ID for your main drive, replace your boot with the full path to the ID. (NOTE: the original /dev/sda is commented out and only there for reference, you don't need to keep it.)

```
#boot = /dev/sda
boot = /dev/disk/by-id/ata-Crucial_CT480M500SSD1_13440956798B
```

Once you have the boot portion set up, you can then move onto the root partitions for your OSes.

LABEL and UUID

To use Labels and UUIDs, you need to use an initrd. It won't work without it. Since PARTLABEL in your lilo.conf isn't supported, if you want to use a label there, you will have to use this method. When building your initrd, you need to make sure you pass the -r option to specify your root partition, and set it to either your label or UUID. You can use the `/usr/share/mkinitrd/mkinitrd_command_generator.sh` script, although, if you're building it for a kernel version that you're not currently running (but have installed), make sure you pass the kernel version you intend to use to the script using the -k option. It will then spit out a mkinitrd command you can use, which you would just need to adjust the -r option.

```
root@darkstar:~# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 4.4.14
#
# mkinitrd_command_generator.sh revision 1.45
#
# This script will now make a recommendation about the command to use
# in case you require an initrd image to boot a kernel that does not
```

```
# have support for your storage or root filesystem built in
# (such as the Slackware 'generic' kernels').
# A suitable 'mkinitrd' command will be:

mkinitrd -c -k 4.4.14 -f ext4 -r /dev/sda2 -m jbd2:mbcache:ext4 -u -o
/boot/initrd.gz

root@darkstar:~# blkid
/dev/sda2: UUID="25a4dafe-bbf2-413f-a60c-8c38efc0a122" TYPE="ext4"
PARTUUID="1ae0ebfe-02"
/dev/sda1: UUID="d21b5f69-6d0c-48fb-b67c-912ebb0fd18e" TYPE="swap"
PARTUUID="1ae0ebfe-01"

root@darkstar:~# mkinitrd -c -k 4.4.14 -f ext4 -r "UUID=25a4dafe-bbf2-413f-
a60c-8c38efc0a122" -m jbd2:mbcache:ext4 -u -o /boot/initrd-4.4.14.gz
OK: /lib/modules/4.4.14/kernel/fs/jbd2/jbd2.ko added.
OK: /lib/modules/4.4.14/kernel/fs/mbcache.ko added.
OK: /lib/modules/4.4.14/kernel/fs/jbd2/jbd2.ko added.
OK: /lib/modules/4.4.14/kernel/fs/mbcache.ko added.
OK: /lib/modules/4.4.14/kernel/fs/ext4/ext4.ko added.
35942 blocks
/boot/initrd-4.4.14.gz created.
Be sure to run lilo again if you use it.
root@darkstar:~#
```

As you'll notice, the command provided by mkinitrd uses `/dev/sda2` as the root drive, so we need to change that to match the UUID of the drive before running the mkinitrd command. By default, it will output the `/boot/initrd.gz` file, but I prefer to call mine based on the kernel version (which is helpful to ensure you don't overwrite a known-good initrd). When including the root partition in your initrd, it can be removed from lilo, since it won't use it (however, I've left it in but commented it out for visual reference).

Once you generate an initrd, you need to reference it in your stanza. I don't have any labels set for my primary drive, so I added an example that is commented out. You can use one or the other. As before, I commented out the original device, but it doesn't have to remain in the conf file.

Instead of changing your original stanza, it might be beneficial to keep it there in case you accidentally screw something up and your system doesn't boot, then you can just select your original config and go and fix things and try it again. Once you have verified your system boots properly with your new stanza, if desired, you can remove the old one.

```
image = /boot/vmlinuz-generic-3.10.17
initrd = /boot/initrd-3.10.17.gz
#root = /dev/sda2
#root = "UUID=23bce2c2-996d-449e-89cc-0e5029cc6d8d"
#root = "LABEL=Mylabelhere"
label = Slack-generic
read-only
```

```
image = /boot/vmlinuz
root = /dev/sda2
label = Slack-default
read-only
```

PARTUUID

Now, as mentioned above, PARTUUID does not require an initrd (although, it works fine if you do use one). However, lilo is old enough that it doesn't have proper support for it, so there is a workaround to get it working. Instead of referencing root like we did above, we need to replace the root option within an “addappend”. This will add anything extra to the initial append line at the top of lilo.conf. Keep in mind your spaces, as it will be placed directly afterwards, so it might be wise to include an extra space at the beginning of the line to ensure it doesn't accidentally combine words (which would likely cause a kernel panic and prevent you from booting). For me, since my root partition is on a drive with an MBR, I'll have the shortened PARTUUID instead of the regular 32 character length of a proper PARTUUID.

```
image = /boot/vmlinuz-generic
addappend = " root=PARTUUID=6f47c81b-02"
label = Slack-partuuid
read-only

image = /boot/vmlinuz-generic
initrd = /boot/initrd-4.4.14.gz
root = "UUID=25a4dafe-bbf2-413f-a60c-8c38efc0a122"
label = Slack-UUID
read-only

image = /boot/vmlinuz-huge
root = /dev/sda2
label = Slack-default
read-only
```

As noted above, PARTLABEL is not supported with lilo.

Finishing Up

Once you have your `/etc/lilo.conf` file updated, we should test it to make sure there aren't any glaring errors. Lilo has the ability to test a config using the -t option.

```
root@darkstar:~# lilo -t
Warning: LBA32 addressing assumed
Added Slack-partuuid  +  *
Added Slack-UUID  +
Added Slack-default  +
The boot sector and the map file have *NOT* been altered.
```

```
One warning was issued.
```

The LBA32 warning is normal with Slackware's `lilo.conf` and can be ignored (however, you can get rid of it by added LBA32 in the Global section of your `lilo.conf`). The main thing we're looking for is that all of our stanzas were added and no errors were issued. If that worked, then we can run lilo by itself to finish the process.

```
root@darkstar:~# lilo
Warning: LBA32 addressing assumed
Added Slack-partuuid  +  *
Added Slack-UUID  +
Added Slack-default  +
One warning was issued.
```

If you got this far, go ahead and reboot. Your system should now no longer rely on device names that can easily change.

Final thing to remember, is your device names can change at any time. As should always be the case, whether or not you're using persistent names, before you do anything that may wipe a drive (e.g. `dd`, `fdisk`, etc), ensure you are using the correct drive name. You don't want to assume your thumbdrive is `/dev/sdc` (because that's what it's always been) before using dd to write that shiny new Slackware64-current iso on there. If your device names have changed, you could be writing that iso to your root partition and take out your whole OS. Use the tools provided above (the various folders under `/dev/disk/`, `blkid`, and/or `dmesg`) to verify what device name you want to use before running the command.

Sources

- Originally written by Jeremy Hansen aka bassmadrigal

howtos, author bassmadrigal