1 How to Reinstall Grub Onto Your MBR

Executive summary: it is fairly easy to recover from certain situations where your machine won't boot the way you want (or at all).

- As discussed in section 2, there are lots of ways the MBR (master boot record) can be damaged or changed in undesirable way. Section 1.1 will tell you how to reinstall or recreate a well-behaved MBR, without having to reinstall everything else.
- Less commonly, the first stage of grub works OK, but later stages fail, due to a damaged grub.cfg file.

Various scenarios that could cause this situation are discussed in section 2, but for now let's concentrate on getting out of the situation.

1.1 Getting the System to Boot

Here is a terse summary. Every step in this section is covered in more detail in section 4.

1. Boot your system using a "live DVD" or (preferably) a "Live Demo" image on a USB drive, perhaps a so-called thumb drive.

2. Open a shell window and become root: `sudo su`

3. Create a directory: `mkdir /x`

4. You need to know which drive partition holds the target system, i.e. the Linux system you want to boot. For clarity, let's discuss things using the shell variables $partition and $drive. An example might be: `partition=/dev/sda6 ; drive=/dev/sda`

If you happen to know, based on experience, where the target system lives, define $partition and $drive accordingly, and

skip to step 8. If you need to figure it out, proceed as follows:

- Get a list of disk drives: `cd /dev/disk/by-label; ls -al`

  With any luck, you will see something like this:

```
drwxr-xr-x 2 root root 200 Mar 26 07:59 .
drwxr-xr-x 6 root root 120 Mar 26 08:00 ..
lrwxrwxrwx 1 root root  10 Mar 26 08:00 emu -> ../../sda9
lrwxrwxrwx 1 root root  10 Mar 26 08:00 linux-root -> ../../sda5
lrwxrwxrwx 1 root root  11 Mar 26 08:00 lnx -> ../../sda11
lrwxrwxrwx 1 root root  11 Mar 26 08:00 more -> ../../sda10
lrwxrwxrwx 1 root root  10 Mar 26 08:00 SERVICEV003 -> ../../sda1
lrwxrwxrwx 1 root root  10 Mar 26 08:00 SW_Preload -> ../../sda2
lrwxrwxrwx 1 root root  10 Mar 26 08:00 usrsrc -> ../../sda7
```

- If necessary, you can get additional information from `cfdisk /dev/sda`. This has the advantage of showing the partition size, along with the partition-type for each partition.
- Identify which partition(s) might plausibly hold the target system's root directory. In the example above, `linux-root` is almost certainly the right answer, but `lnx` is a semi-plausible alternative.
- To make sure, mount each plausible partition and take a look at it.

```
mount /dev/sda11 /x
ls -al /x/boot/vm*
ls: cannot access /x/boot/vm*: No such file or directory
```

  That tells us that sda11 is definitely not the right answer. So let's try again:

```
umount /x            # unmount previous hypothesis

mount /dev/sda5 /x
ls -al /x/boot/vm*
-rw-r--r-- 1 root root 4275712 May 30  2013 /boot/vmlinuz-2.6.39.4
-rw-r--r-- 1 root root 4678720 Aug 25  2013 /boot/vmlinuz-3.10.9
-rw-r--r-- 1 root root 4599824 Mar 23 18:45 /boot/vmlinuz-3.11.3+
-rw-r--r-- 1 root root 4599824 Mar 23 18:43 /boot/vmlinuz-3.11.3+.old
```

5. When you find the desired partition, leave it mounted on the mountpoint `/x`.

6. Define $partition and $drive accordingly. Example: `partition=/dev/sda5 drive=/dev/sda`.

7. If not already mounted: `mount $partition /x`

8. Reinstall grub: `grub-install --root-directory=/x $drive`

   Beware: You want to install grub on the *drive* (e.g. /dev/sda). If you install it on the partition (e.g. /dev/sda6), the grub-install program won't complain, but the results won't be what you wanted.

That's probably enough to get you going. Shut down the Live Demo system, eject the DVD if any, and reboot in the normal way from your favorite drive (/dev/sda in the example).

1.2 Status Check

Let's assume you have already carried out the steps in section 1.1. Various possible next steps include:

- The system boots up fine. See section 1.4 for some housecleaning suggestions. Then resume normal operations.
- Grub prints a normal-looking menu, but when you try to boot Linux, the boot does not succeed. This might be a problem with the grub configuration. Try rebuilding the grub.cfg file according to the instructions in section 1.3.
- Grub doesn't even print a menu, but instead prints "grub rescue" prompt. This might also be a problem with the grub configuration. Try rebuilding the grub.cfg file according to the instructions in section 1.3.
- Sometimes it isn't a grub problem at all. For example, if you build a 64-bit kernel and install it on a machined with 32-bit utilities, it will not boot successfully, and there is nothing grub can do about it. You have to install a correct, consistent set of software. You may need to reconfigure grub also, but that happens *after* you have solved the other problems.

1.3 Rebuilding the Grub Configuration

9. If the Live Demo system is not already running, restart it.

10. Back up the existing grub control file, namely grub.cfg: `cd /boot/grub ; cp grub.cfg grub.cfg#1`.

Note that we are assuming that grub version 2 is being used. It has been several years since anybody used grub version 1, so I'm not going to worry about it.

11. Assuming the Live Demo system has a /boot directory, move it out of the way: `mv /boot /xxxboot`. Then put the target system's /boot directory in its place: `ln -s /x/boot /`.

12. We are about to create a hacked version of the `grub-mkconfig` file, so first make a copy: `cd /usr/sbin/ cp grub-mkconfig grub-hackconfig`.

Fire up the text editor: `vi grub-hackconfig`.

Find the line that says `GRUB_DEVICE=...` and change the last thing on the line, so that it probes `/x` rather than `/`.

Similarly, find the line that says `GRUB_DEVICE_BOOT=...` and change the last thing on the line, so that it probes `/x/boot` rather than `/boot`. Here is the diff:

```
--- grub-mkconfig       2014-03-26 13:51:30.649215458 -0700
+++ grub-hackconfig     2014-03-26 13:47:31.895892874 -0700
@@ -136,11 +136,11 @@
 fi

 # Device containing our userland.  Typically used for root= parameter.
-GRUB_DEVICE="`${grub_probe} --target=device /`"
+GRUB_DEVICE="`${grub_probe} --target=device /x`"
 GRUB_DEVICE_UUID="`${grub_probe} --device ${GRUB_DEVICE} --target=fs_uuid 2> /dev/null`" || true

 # Device containing our /boot partition.  Usually the same as GRUB_DEVICE.
-GRUB_DEVICE_BOOT="`${grub_probe} --target=device /boot`"
+GRUB_DEVICE_BOOT="`${grub_probe} --target=device /x/boot`"
 GRUB_DEVICE_BOOT_UUID="`${grub_probe} --device ${GRUB_DEVICE_BOOT} --target=fs_uuid 2> /dev/null`" || true

 # Filesystem for the device containing our userland.  Used for stuff like
```

13. Update the grub control file, using the command `./grub-hackconfig -o /x/boot/grub/grub.cfg`. This does a lot of work, in order to produce a new version of the `/x/boot/grub/grub.cfg` file.

Now you really should be ready to shut own the Live Demo system, remove the DVD if any, and reboot in the normal way.

1.4 Housecleaning and Follow-Up

The procedures in section 1.1 were meant to get the system functioning again as quickly as possible. Now that the system is up and running, so that the time pressure is off, we can do some housekeeping:

A. Optional: You may want to make sure your copy of the software is not corrupted: `apt-get install --reinstall grub # (optional)`
B. You should make a backup of the MBR as described in section 3.1.
C. Highly recommended: Rebuild the grub configuration file: `update-grub`
D. Install the latest and greatest grub in the MBR: `grub-install --recheck /dev/hda`

In ideal situations, the work described in this section doesn't accomplish much, because it duplicates the work done in section 1.1 and section 1.3. However, consider the situation where the Live Demo system you used to restore the MBR is using a different version of grub. Maybe one system is out of date, or maybe just exercised the option to use a different version. This is your chance to install the grub version that your system thinks should be installed. If you don't do this, you risk having some ugly problems later.

2 Scenarios and Alernatives

There are several scenarios that can lead to an MBR being overwritten or otherwise rendered unsatisfactory. Examples include:

- On a dual-boot system, every time you install (or reinstall) Windows, it will almost certainly overwrite your MBR. See section 2.1.
- A failed upgrade can leave grub in a bad state. In particular, if the system was using Grub Version 1 before the upgrade and wants to use Grub Version 2 afterwards, sometimes things get confused. I've seen it happen.
- Viruses and other malicious software are fond of overwriting the MBR.
- et cetera.

2.1 Dual Boot

Suppose you have a *dual boot* system, i.e. one that sometimes boots Linux and sometimes boots Windows. Every time you install (or reinstall) Windows, it installs its own boot loader into the MBR. This is a problem, because the MS boot loader will not load anything except the MS operating system ... in contrast to grub, which will happily allow you to boot almost anything: Linux, memtest86, various MS products, et cetera.

Some folks recommend installing MS before installing Linux, so that the Linux installation process will set up the MBR for you. This is fine as far as it goes, but it is not always possible. For instance, sometimes it is necessary to *reinstall* or *upgrade* the MS stuff, days or months or years after Linux was installed.

The grub-reinstall procedure described in this document takes only a few minutes, so feel free to install MS after Linux if you find it necessary or convenient to do so. MS will trash the MBR, but you can restore it using the techniques described here.

3 Backing Up and Restoring the MBR

3.1 Backup

It never hurts to make backups of sector 0.

```
dd if=/dev/sda of=host1-sda.mbr count=1
```

- If you have two or more Linux systems, use system "1" to store the backups pertaining to system "2" and vice versa.
- If you have only one system, store the backups on floppy ... and don't forget where you put the floppy.
- Feel free to keep *another* copy of sector 0 on the same drive as the sector you are backing up. This is useful in cases where part of sector 0 is messed up but the partition table remains correct. It is useless in cases where the partition table trashed.

3.2 Restore

Keep in mind that sector zero contains both the stage-0 boot code and the primary partition table. Therefore, before restoring the boot sector, you have to make a decision:

- In the scenario where something trashes sector 0 including the partition table, then you want to restore the whole thing. This

can rescue from what would otherwise be a very bad situation.

```
dd if=host1-sda.mbr of=/dev/sda count=1
```

- In the scenario where the partition table is not trashed, and has possibly changed since you backed up the MBR, you want to restore the boot code without disturbing the current partition table. You need to splice the backed-up boot code onto the current partition table *before* writing anything to sector 0. The procedure is:

  Keep a copy, just to be safe: `dd if=/dev/sda of=damaged.mbr count=1`

  Grab the good boot code from backup: `dd if=host1-sda.mbr bs=1 count=444 > new.mbr`

  Tack on the current partition table: `dd if=/dev/sda bs=1 skip=444 count=68 >> new.mbr`

  Write to disk: `dd if=new.mbr of=/dev/sda count=1`

4 Details

Some discussion of the MBR and the basic boot process can be found in reference 1.

4.1 Live Demo Images

- Ubuntu: The Ubuntu Live USB drive (or DVD) that you used to install Ubuntu also serves as a nice Live Demo image, suitable for many purposes including the grub reinstallation process described here. So be sure to keep that USB drive (or DVD) handy. If you need to download a new copy, see reference 2.
- Debian: The usual Debian install disk is not, alas, a fully-featured Live Demo. A rundown of the various Debian Live images can be found in reference 3.
- Slackware: RIP (reference 4) is a Slackware Live Demo, suitable for tasks such as grub reinstallation.

4.2 Superuser Privileges

We now discuss the step `sudo su`

For good reasons, when you fire up a typical live CD, you are logged in as an ordinary user, not the superuser.

You can exert superuser privileges on a command-by-command basis by prefixing each command with "sudo" ... but since every command we are about to do requires superuser privileges, it is easier to just become superuser once and for all by saying `sudo su`

4.3 Mountpoint

We now discuss the step `mkdir /x`

This creates a new empty directory named `x`. The name is arbitrary, made up just for this purpose. You could use any other name if you wanted, so long as you used the name consistently in all steps in the grub-reinstall procedure ... but `x` is as good as any. It's just some empty directory. It serves the following purpose: In a moment we are going to want to mount a filesystem. Linux mounts things by mounting them onto a directory. The newly mounted filesystem has to attach to the rest of the filesystem somewhere, and Linux uses a pre-existing directory as a point of attachment.

4.4 Mounting Your Linux Partition

We now discuss the step `mount /dev/sda6 /x`

Not much to say, really. If you want the operating system to treat your partition as a collection of files and directories (as opposed to a bucket of bits) you need to mount it.

4.5 Grub Installation

We now discuss the step `grub-install --root-directory=/x /dev/sda`

The `--root-directory=/x` option tells grub where to look for the grub directory *during the installation process*. The grub directory is `/x/boot/grub` on typical distributions such as Ubuntu and Debian, but may be `/x/grub` on some *bsd setups.

The `grub-install` program uses the grub directory in several ways during the installation process. Among other things, it goes there to read the `device.map` file. It also goes there to write the `core.img` file. A new `core.img` file gets written each time you run `grub-install`.

Keep in mind that the Unix file system is essentially a graph (in the sense of graph theory) with edges and nodes. The edges are the paths, i.e. directory names and file names. The nodes do not have names. The nodes are where the data is stored. So: the inode of interest will be reached by the path "`/x`" during the installation process. Grub assumes this inode will be reached by the simple path "`/`" later, when the system on /dev/sda6 is actually booting and running.

The idea that the same inode could be reached by one path now and a different path later makes perfect sense if you think about it the right way. The `grub-install` program understands the distinction between the two, which is what makes it possible to reinstall grub using the easy procedure described in this document.

This distinction is, alas, not well documented. You could read the grub manpage all day and not learn anything about this distinction. The `grub-install --help` message says

```
--root-directory=DIR    install GRUB images under the directory DIR
                        instead of the root directory
```

which seems somewhere between incomprehensible and self-contradictory. Is DIR the root directory (as suggested by the equation root-directory=DIR) ... or is DIR used "instead of the root directory" (as stated in the explanatory message)? Gaaack.

5 Using Grub Commands Directly

I hope you never need to know this. Usually the procedures described in section 1.1 make this unnecessary.

Imagine a scenario where grub is installed in the MBR correctly, but the grub configuration files are messed up, so all you get is the `grub>` prompt (rather than a menu of kernels that can be booted). Further imagine that you can't fix it using the methods described in section 1.1.

You may be able to recover using the following procedure:

- At the `grub>` prompt, type `root (hd0,<tab>`

  This will give you a listing of all the partitions on the hd0 drive, along with their UUID, filesystem type, and modification date.

  If hd0 turns out to be not the drive you want, try hd1 and so on.

- Pick the partition you want, say #2, and issue the complete command: `root (hd0,2)`
- At the `grub>` prompt, type `linux /boot/vml<tab>`

  This will give you a listing of all the filenames in the boot directory that start with “vml”. (If your kernel isn’t named vmlinuz-something, adapt these instructions accordingly.)

- Pick the kernel you want, and issue the complete command, e.g.: `linux /boot/vmlinuz-2.6.35.10 root=/dev/hde3`

  Note that you generally have to add the root=... option to the linux command line.

  Beware that the way grub numbers disk drives {hd0, hd1, hd2, etc.} may be different from the way linux does it {sda, sdb, sdc, etc.} ... and the difference is not systematic. I have one system where hd0 corresponds to /dev/hde/. This is commonly an annoyance on systems that have a mixture of SATA and PATA drives.

  The numbering of partitions is also different, but the difference is systematic: grub numbers them starting from 0, while linux numbers them starting from 1, so grub partition (...,2) coresponds to linux partition /dev/...3 and so on.

- At the `grub>` prompt, type `initrd /boot/init<tab>`

  This will give you a listing of all the initrd files. Pick the one that corresponds to your kernel, and issue the complete command: `initrd /boot/initrd.img-2.6.35.10` or whatever.

- Issue the `boot` command. The kernel should boot.
- If the kernel panics because it could not mount the root fs, it means you guessed wrong about the root=... command-line argument. Maybe it is /dev/hda3 or /dev/sda3 or /dev/sde3. However ...
- Remember that the kernel needs to know the root drive twice, once when it is reading the initrd (initial ramdisk), and once again when it is starting the system for real. I have seen situations where the drive is named differently in the two cases, in which case any drive name you pick is going to be wrong in one context or the other, and the system will not boot correctly.

  The only way to handle this case is to refer to the disk by its UUID, using a construction of the form root=UUID=4240ce68-802b-4a41-8345-543fad0ec20f

  That is an obnoxious amount of typing, but with any luck you only have to do it once.

Grub will tell you the UUID; see the first item in this list.

- Once the system is booted, clean up the mess using the methods described in section 1.4.

6 References

1. Wikipedia Article: “Master Boot Record” http://en.wikipedia.org/wiki/Master_boot_record
2. Download Ubuntu CD (installer == live CD) http://www.ubuntu.com/GetUbuntu/download
3. A summary of Debian Live CDs http://wiki.debian.org/LiveCD
4. RIP – Recovery Is Possible http://www.tux.org/pub/people/kent-robotti/looplinux/rip/index.html

[Contents]

Search