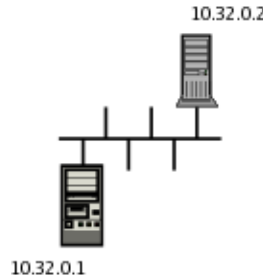


LF Tip:Cloning entire PCs over the network



by Gerrit Renker
 <gerrit.renker(at)gmx.de>

About the author:
 Obtained a computer science degree in 2001.



Abstract:

Often the problem arises that you have to replicate data from one computer to another. A safe, easy-to-do and effective method is to use network-based cloning as described in this howto.

Scenario

While cloning of animals ("Dolly the sheep") and even human embryos is a much-debated and dubious research area, there is no doubt that some knowledge in cloning computer machines is not only less harmful (if done properly) but also useful for the evolution of one's carefully written configuration scripts and settings. Due to Moore's Law and the fast progress in computer manufacturing, one is almost certain to encounter at least one cloning scenario during the lifetime of using a distribution – be it that the home PC is to be replaced by a laptop or just that a faster processor is on the market and there is enough money to buy a new computer. The task is to *take the entire filesystem* of computer A and put it to work on computer B. There are two ways of doing it; the one which is not described in this howto is to open the cases, physically swap hard drives among computers and do the copying locally on one machine. This is often not possible – opening the case cover often will result in losing the warranty – and also has dangers, since an inexperienced person can physical and electrically damage the hardware. Even more so, I once lost disk data this way due to a buggy low-level program. The other approach requires that both PCs have network cards (which even in home systems is now frequently the case), is much safer and described here.

The underlying idea common to all methods described below is to establish a network connection between the "source" computer (which is to be cloned) and the "target" computer (the clone). This is straightforward if both are plugged into a hub, otherwise you can connect the network cards via a *crossover cable* (normal straight cables can not be used). For the target PC, a Live-CD (such as Knoppix or LNX-BBC) or a minimalist installation is needed such that the network card is operable and `ssh` and/or `netcat` can be used. There are even some floppies which allow this (although in my case `tomsrtb` hung on initialising the network card). If you intend to install another (fresh) distribution anyway, this is an easy alternative. Both computers need to be configured with IP addresses on the same network such that they can "talk" to each

other, as shown in the above picture.

Possible methods

With given basic set-up there are several ways of performing the act of cloning:

- binary copy via `dd`
- `tar` / `cpio` pipes
- `rsync`
- `dump` and `restore`

The first is complicated if not impossible if your hard drives are not exactly of the same type and geometry. It is great for things such as copying iso images (`dd if=/dev/cdrom of=the.iso`), or floppies – [here \(diskcopy shell script\)](#) is an example of a `diskcopy` script using `dd`. The other disadvantage of the `dd`-based approach is that the unoccupied space is also copied, thus taking needless time. The `tar` and `cpio` pipes take a very long time (up to several hours) and have several problems. For instance, there are restrictions on the filenames and use of symlinks, choking on files in `/dev`, and the like. I would generally not recommend this approach for cloning. If you do have different filesystems on the source and the target, **`rsync(1)`** is probably the best choice. It merely requires that `ssh` is running and effectively transmits files due to an efficient, built-in protocol. It does even have a `-D` switch for device files, as well as many more options to cater to most practical scenarios. It is a very useful tool for daily backups, mirroring and the like, the man-page with its many examples is well-worth studying. An example of cloning via `rsync` is presented in [1].

Here, we use the method via `dump` and `restore` which amounts to re-creating the entire filesystem. It is fast, effective and delivers the desired results with minimal effort, hence ideal for full cloning. I actually had to do the entire cloning procedure twice, since the target PC was called back and replaced. In both cases absolutely no problems were encountered and lead out a functional, bootable clone within roughly one hour of copying gigabytes. This approach requires that both target and source PC feature the same filesystem. We assume that this is `ext2` or `ext3`, since it is currently the most widely used kind (see notes [below](#)).

Setting up ssh

Once you have configured a system with a minimal installation or have a Live-CD running, the next step is to set up `ssh` (if you are not using `netcat` for the transfer as detailed further below). This requires that the source PC has `sshd` (the secure shell daemon) running. Check `/etc/init.d/` if unsure. On the target PC, type (as root)

```
ssh-keygen -t rsa
```

To keep things simple, do not enter a passphrase. The public key is now in the file `/root/.ssh/id_rsa.pub`. Copy this file to your source PC via

```
scp /root/.ssh/id_rsa SourcePC:/tmp
```

where `SourcePC` is the IP address of the source PC. When asked whether you are sure, enter a full "yes" ("y" alone sometimes does not work). You are still being prompted for the root password on the source PC. Now add the target PC as a trustworthy network node by issuing

```
cat /tmp/id_rsa.pub >> /root/.ssh/authorized_keys
```

on the source PC. To check whether everything is ok, repeat the above copy command on the target PC. You should no longer be prompted for a password.

Creating a file system on the target PC

The first step is always to partition the hard drive on the target system and then to create the `ext2/ext3` filesystem. The latter is the preferable journalling variant and is enabled simply by setting the `-j` (journalling) option in `mke2fs` (requires to have `ext3` support in the kernel). You can even convert an `ext2` system to `ext3`, see **tune2fs (8)**. So let's say on the source PC we have the following configuration:

Filesystem	Size	Used	Use %	Mounted on
/dev/hda3	2.7G	552M	22%	/
/dev/hda5	7.8G	1.6G	22%	/usr
/dev/hda7	6.3G	1.7G	28%	/usr/share
/dev/hda8	3.4G	601M	19%	/home
/dev/hda12	5.3G	1.9G	37%	/opt
/dev/hda1	587M	70M	13%	/var/backup

I recommend to always do some kind of partitioning. If not, then any failure in using the filesystem or a hard-drive knock-out of just some sectors will destroy *all* your data. And according to Murphy's law, this is sure to happen if no precautions are taken in form of using different partitions instead of a single monolithic one. I had such a case recently with a funny kernel and had I not partitioned the drive, I would have lost all my data along with the munched root file system. The above actually shows that `/usr` was growing too big, therefore `/usr/share` had been added. Time for a bigger hard drive.

On the target PC, fire up `parted` (recommendable) or your favourite partitioning program (Qtparted is a nice GUI-variant, said to be a Partition Magic clone). Create partitions which are all at least as big as the ones on the source PC. Don't forget the swap partition. After saving the partition table, put a filesystem on all newly created partitions, using

```
mke2fs -j -L <label> /dev/xxx
```

where `xxx` is the partition name and `<label>` a label string. I usually just use things like `"/usr"` as labels (you will see that at boot-time). You can set various things via **tune2fs (8)**, such as the regular file system check interval.

Transfer the filesystem

First you need to mount all the newly created partitions. We start with the root filesystem ("/") and mount the remaining directories as we go along. It is perfectly possible to condense two partitions of the source PC into a single one on the target, in fact this is what we are going to do with `/usr/` and `/usr/share` in the above example. So mount your future root filesystem using

```
mount /dev/xxx /mnt
```

When cloning, it is necessary to `chdir` into the target directory

```
cd /mnt
```

Now the network bit, on the target PC type

```
ssh sourcePC 'dump -0 -f - /' | restore -r -f -
```

where `targetPC` is the IP address of the target PC. The options mean "-0" for a full backup, "-f -" tells it to use `stdin/stdout` as file descriptors and "-r" instructs restore to re-create the filesystem being piped over the network on the target PC. For more options see **dump (8)** and **restore (8)**. Below you see the output for transferring the root filesystem.

```
$ ssh 10.42.3.42 'dump -0 -f - /' | restore -r -f -
DUMP: Date of this level 0 dump: Tue Feb 22 15:50:12 2005
DUMP: Dumping /dev/hda3 (/) to standard output
DUMP: Label: debian
DUMP: Writing 10 Kilobyte records
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 547312 blocks.
DUMP: Volume 1 started with block 1 at: Tue Feb 22 15:50:14 2005
DUMP: dumping (Pass III) [directories]
DUMP: dumping (Pass IV) [regular files]
DUMP: Volume 1 completed at: Tue Feb 22 15:51:43 2005
DUMP: Volume 1 546590 blocks (533.78MB)
DUMP: Volume 1 took 0:01:29
DUMP: Volume 1 transfer rate: 6141 kB/s
DUMP: 546590 blocks (533.78MB)
DUMP: finished in 89 seconds, throughput 6141 kBytes/sec
DUMP: Date of this level 0 dump: Tue Feb 22 15:50:12 2005
DUMP: Date this dump completed: Tue Feb 22 15:51:43 2005
DUMP: Average transfer rate: 6141 kB/s
DUMP: DUMP IS DONE
```

Restore always creates a file `restoresymtable` which can be removed once you are certain that no errors occurred during the file system restoration. When done with the root filesystem, we now proceed with each mounted sub-filesystem, starting with `/usr` (assuming your current work directory is the root of the future filesystem).

```
mount /dev/xxx ./usr
```

```
cd ./usr
```

```
ssh targetPC 'dump -0 -f - /usr' | restore -r -f -
```

The `mount-cd-dump/restore` cycle is now repeated for all the directories you may have. With regard to `/usr/share` (which on the source PC had its own partition), you can, after the above step, simply `chdir` into `./usr/share` (note the ".") and then repeat

```
ssh targetPC 'dump -0 -f - /usr/share' | restore -r -f -
```

Restore only complains if files already exist in the filesystem being restored, so when putting two different partitions of the source PC into a single one on the target PC there is no problem. Cloning an entire PC took just about one hour with `ssh` and 100MB network cards (crossover cable a bonus).

Note: To dump a filesystem, it need not necessarily be mounted. You can also pass a *partition name*, such as `/dev/hda6`, instead of the directory name of a mounted partition.

Alternative: netcat

An alternative to use instead of `ssh` is `netcat` (1), which is abbreviated as `nc`. Netcat is a simple-to-use TCP/IP client-server swiss army knife which allows to create a pipe over the network. The above examples are then merely modified as follows. We assume that the partition mounted under `/var/backup` is to be transferred via `dump/restore` from the source PC to the target PC.

On the receiving end (*target PC*) create a listening instance of `netcat` via `-l` which pipes its output to `restore`.

```
nc -l -p 2000 -q 1 | restore -r -f -
```

On the *source PC*, create another instance of `netcat` which takes its input from a pipe where `target-IP` is the IP address of the target PC.

```
dump -0 -f - /var/backup | nc <target-ip> 2000
```

The `-q` option is supposed to stop `nc` after receiving end-of-file, but I had to terminate it manually in my case. Would recommend using `ssh` anyway.

Cleaning up

Congratulations, if thus far successful you basically have a cloned system. Now it remains to turn this clone into a workable PC. The first thing to do is to update `/etc/fstab` with the new settings, otherwise you will not be able to use the cloned partitions. If the IP address changes, this also needs to be updated (`/etc/hosts`, `/etc/network/interfaces` under `debian`). Next important thing is the *boot configuration* which in almost all cases will want updating. With `lilo`, edit `/etc/lilo.conf` (in particular the `root=...` option) and then run `lilo -v`. Under `grub`, edit `/boot/grub/menu.lst` (or `/boot/grub/grub.conf` depending on which one is the symlink) and then enter `grub`,

```
grub> root (hd0,xxx)
... filesystem is ...
grub> setup (hd0)
... lots of output here
grub> quit
```

or run `grub-install /dev/xxx` where `xxx` is your hard drive. Here, check both for the `root (hdn, xx)` and the appended `root=/dev/xxx` settings.

In the likely case of now having better hardware in the newly cloned PC, you may want to update the *settings for your custom-kernel*. If you are using systems which come with lots of pre-configured modules (such as RedHat, SuSe, Mandrake, Fedora ...) don't worry, it is quite likely that there will be a suitable module. If not, `lspci -vv` and kernel compilation as usual and described elsewhere. If your video card is different now, update `/etc/X11/XF86Config-4` (or `xorg.conf` under RH/Fedora) to reflect this, otherwise you will get no output. If possible use the graphical tools for setting up X by booting in runlevel 3 if you have such tools. Under `debian`, some investigation is necessary, I was lucky to find out that the driver changed from `r128` to `radeon` and that did it.

Other systems

This howto explained the cloning procedure for `ext2/ext3` filesystems. Much similar commands can be found on many other Linux systems. For instance, several Unices such as FreeBSD, HP-UX, IRIX also provide `dump/restore` commands; in Solaris this is called `ufsdump/ufsrestore`. There are filesystems which do not offer filesystem dump functionality, e.g. ReiserFS. Here it would suggest itself to use `rsync`. See [1] for a report of successfully using `rsync` to clone a Linux system.

References

[1] "*Replicating a Linux System – Yet Another Method.*" Ben Okopnik, Linux Gazette Issue 83, October 2002.

<p><u>Webpages maintained by the LinuxFocus Editor team</u> © Gerrit Renker "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org</p>	<p>Translation information: en --> -- : Gerrit Renker <gerrit.renker(at)gmx.de></p>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------

2005-03-27, generated by lfparsr_pdf version 2.51