HOW TO CLONE A LARGE NUMBER OF PCs

© András Keszthelyi
(Budapest Tech – Polytechnical Institution)

kea@turul.banki.hu

Computer sciences are taught in all kinds of schools from elementary grade to universities both in theory and in practice. Schools usually have computer labs to serve this education activity. The technical management of these labs needs very different type of activities. One of them is to install or re-install the computers, which can be a very dully and time-consuming activity with the possibility of human mistakes. In this article I show two methods for two different situations to automate this process.

Keywords: Linux, computer lab management

In our school, the Budapest Polytechnic, there is a department computer lab with 20 identical personal computers in it, and only one additional cd/dvd rom in the 20th machine. Sometimes it is necessary to install or re-install an operating system and its applications on these machines. Istalling individually on each of twenty machine is not only a very slow and dull process but can result in less or more differencies among the computers. So the task is to install these machines with the simplest possible way, i.e. with the least human interaction by "copying" the installed system from one machine to another. This is the so-called cloning. There are some softwares to manage this task but I would like to solve the problem without any special softwares with only the standard utilites of the operating system. My solution is based on Linux.

Our computers are set up with Linux operating system. This decision has some advatages in our situation. Our students can not only see but try it out personally. From the point of view of the maintainer of the computer lab the possibility of the remote and batch mode maintaining is an important circumstance as well as the stability of the system (in an environment full of students:). Having lawful softwares in the whole lab is also very important. It is not widely known that the Microsoft Campus Licence is an *upgrade* licence so it needs a legal copy of a previous Windows version.

The hardware configuration is the following: P-IV processors of 1,5 GHz clock speed, 256 MB RAM, hard drives of 40 GB in each computer. The hard drives are divided into two partitions: the first (/dev/hda1) is a six GB partition for the operating system and the applications while the rest for the working place. The local network is a 100 Mbit/sec ethernet.

At this point two cases can be distinguished. We are in a lucky situation with the totally identical hardwares but the method to be made known later is good to handle the case of different hardwares, too.

First case: Absolutely identical hardware and Windows

If we have the very same hardware configuration even Windows operating systems can be cloned as well. This method needs more time than the other. The basics of the method is to make a disk-copy between two computers, i.e. to make a sector by sector copy of the system (boot) partition.

The first step is to make the source. Operating system must be installed and prepared (e.g. with sysprep) on one computer. Suppose that the primary master hard disk is used. The free space on the system partition should be zeroed in order to have better compression rate. This can be done by writing zero bytes (or spaces) to a file until the disk is full then by deleting this file.

The second step is to prepare the second partition. Boot Linux from CD on the newly installed Windows machine. The second partition is now empty yet. Make a suitable filesystem (e.g. Reiser journalling filesystem) on it:

```
mkreiserfs /dev/hda2
```

Mount the second (writeable by Linux) partition to an existing mount point:

```
mount /dev/hda2 /mnt/hda2 -t reiserfs
```

At this point we have a fully prepared, ready to clone Windows partition (/dev/hda1 under Linux) and a Linux partition which is able to store the partition image.

Third step: to make the partition image. This will take some time, because the whole partition should be read and compressed into one (huge) file. To guarantee the identical (free of human mistakes) partitioning of the other computers we save the master boot record as well. Be careful and do remember the difference between hda, hda1 and hda2.

```
dd if=/dev/hda1 bs=512 | gzip -c9 > /mnt/hda2/winimage.gz

dd if=/dev/hda of=/mnt/hda2/mbr.dat bs=512 count=1
```

The two files in /mnt/hda2/ can be saved for later use, of course. For the following step suppose that they are on an nfs-capable Linux box, either on the original, cd-booted machine or on a real Linux server. The Linux machine should be configured to export /mnt/hda2 for nfs (network file system). Because of the large amount of data it is better to have our Linux nfs server on the same subnet as the machines to be cloned. The very basics of making nfs server and client under Linux is described later.

Fourth step: cloning the other machines. Supposing that the other machines have no CD/DVD drives boot Linux on them one by one from floppy disk(s). If one has CD/DVD-ROM or a USB-boot-capable BIOS in each machine that makes the situation more comfortable. The very basics of making boot floppies under Linux is discussed later. After the machine to be cloned has been booted it must be act as an nfs-client and the following commands must be performed:

```
ifconfig eth0 a.b.c.d
```

where a.b.c.d is the IP address of the floppy-booted machine (e.g. 192.168.88.19). At this point the portmapper program should be able to start:

```
rpc.portmap
```

Supposing that a.b.c.e is the IP address of the Linux nfs server machine (e.g. 192.168.88.2) the directory containing the partition image can be mounted via nfs:

```
mount a.b.c.e:/mnt/hda2 /mnt -t nfs
```

Next we must make the needed partitions on the new machine by the help of the saved mbr data:

```
dd if=/mnt/mbr.dat of=/dev/hda
```

After this we should run the fdisk program manually and quit with the "w" command in order to force the system to re-read the partition table. At this point we have the two partitions we need, the two partitions with the very same size as that of the original, source machine. Of course these are empty partitions even without a filesystem. We do not need to create any filesystems on the first partition because the partition image contains that as well. The second partition is not needed for the cloning itself, after the first boot of the cloned machine it can be formatted and used as needed.

The command

```
gzip -dc /mnt/winimage.gz | dd of=/dev/hda1 bs=512
```

copies the pre-installed system to the first partition. It takes some (long) time depending the size of the image, the network bandwidth and the processor performance (because all the data should not only be transferred via the network but be decompressed as well) and the performance of the hard drives. In the case of an old 10 Mbit/sec network the limit is the bandwidth while in the case of a 100 Mbit/sec network the the limit may be the hard drive performance. In the latter case the hdparm utility can help (see forward).

When the last gzip command finishes the newly cloned computer is ready to reboot. Do not forget to remove the floppy disk! After rebooting the sysprep-ed Windows must start. I successfully used this method for cloning the computers of my colleagues before.

Second case: Different hardware and Linux

If we use Linux operating system we are in a better situation because a quicker cloning method can be used even if our computer hardwares are different. The absolutely minimal requirement is to have a kernel that can boot on all our machines to be cloned.

We can clone Linux even by copying filesystem itself. This solution takes less time than the first because there is no need to write the whole partition to the end.

We must install Linux on the first computer, of course. There is no need to make other pre- or post-install steps. Then rebooting this machine from CD we can make the "image" file for the cloning. Supposing the same partition scheme as in the first case we have our newly installed Linux operating system and its applications on /dev/hda1 while we have /dev/hda2 as an empty (and large enough) partition for the "image" file.

We can make a compression of the whole system partition by the commands:

```
cd /mnt/hda1
tar -czf /mnt/hda2/hda1files.tgz ./*
```

Since we put (and compress) together all the files of the system partition we ought not to care about the occupied and free disk space. We prepare the nfs-export and boot the other machines just as the same manner as in the first case, but we need three more things to care about.

We must be able to make the empty filesystem on the machines to be cloned because we copy not a whole partition but a filesystem hierarchy. We should run lilo after the decompression but we have no guarantee that the files needed for booting the kernel are in the same sectors as in the source machine. The third thing is to run hdparm at the beginning to improve the speed.

These programs can not be placed on the rootfs floppy because there is not enough space on it. Having placed them on the nfs-exported /mnt/hda2 directory I got only error messages about different or unlocatable dynamic system libraries. I could solve this problem by compiling and linking these utilities from source statically. To make static executables I appended "-static" to the CFLAGS and LDFLAGS variables in the Makefile of each utility. The statically linked programs can be placed in the nfs-exported directory and they run without any problems (at least without problems of the dinamically linked libraries).

The first steps are the same as in the first case. Booting from floppies, setting up the network card, starting the portmapper and mounting our source via nfs.

After completed these steps we should improve the hard disk performance by the hdparm utility. In my case:

```
/mnt/hdparm_static -c1 -d1 -u1 /dev/hda
```

ATTENTION! BE CAREFUL! It must be tested previously what parameters our hard drives can tolerate.

After partitioning the local hard disk just in the same mode as in the first case we must make the filesystem on it:

```
/mnt/mkreiserfs_static /dev/hda1
```

where mkreiserfs_static is the name of the statically linked mkreiserfs utility given by me.

Mount the new partition to a new mount point then decompression can be started:

```
mkdir /mnt1
mount /dev/hda1 /mnt1
cd /mnt1
tar -xzps --same-owner -f /mnt/hda1files.tgz
```

Lilo should be run at the end (lilo_static is the name of the statically linked lilo utility given by me):

```
/mnt/lilo_static -v -r /mnt1
```

Unmount /dev/hda1 and the remote nfs export, reboot the machine

(remove floppy!) and after some little corrections the newly cloned Linux computer is ready to use. These corrections are the following: changing the hostname in /etc/HOSTNAME, the IP-address in /etc/rc.d/inet1.conf (in my case, Slackware distro, so the path and name of this file may vary).

I used this method in our computer lab the other day, also successfully. My situation in the field of the hardware differencies can be considered very good, for I have only two differencies: one of the computers has a ps/2 mouse instead of an usb one and another machine has a different kind of monitor than the others. So after a full re-install I should change some (little) configuration settings only on two pc-s.

Basics of nfs

I think that if one feels himself or herself capable of trying out these methods he or she must have at least a little Linux experience. Supposing this little experience I will not discuss all the aspects of making nfs servers and clients as I did not discuss all the possible problems of the above methods. A brief summary may give the main guidelines.

Kernel must be able to handle nfs file system. On the nfs server at least the following commands must be run:

```
exportfs -r
rpc.portmap
rpc.nfsd
rpc.mountd
```

and the contents of the /etc/exports file is the following:

```
/mnt/hda2/kalyha
192.168.88.0/255.255.255.0(ro,no_subtree_check)
/mnt/hda2/kalyha 127.0.0.1(ro,no_subtree_check)
```

Because the floppy-booted small system has very few tools for tracing possible errors one can try out the nfs-export on the nfs-server (mount localhost:/mnt/hda2 /mnt/aux -t nfs) or at least the command "rpcinfo -p" should produce a list -- or error messages. The possible warning message "mount version older than kernel" on the client machines can be disregarded.

Basics of boot-floppies

We need two error-free floppy disks. One for the kernel and the other for the minimal root filesystem. If we compiled our own kernel before the kernel floppy is not a serious exercise. Simply say "no" in the kernel config for all unnecessary options (sound, video, etc.). The options must be compiled in: support for nfs and for the filesystem of the root fs in our example for reiserfs. In addition the appropriate driver for our network card(s) and initrd support is needed as well.

After having compiled the kernel (make menuconfig; make -s dep; make -s bzImage) we have a bzImage in the [kernel source]/arch/i386/boot directory. If the size of this file is bigger than the size of the floppy we are wrong and must repeat the compilation without more unnecessary options.

Kernel should "know" that the root filesystem must be loaded from floppy disk and that it should prompt for floppy change.

```
rdev bzImage /dev/fd0
rdev -R bzImage 0
rdev -r bzImage 49152
```

After this step the newly compiled kernel can be copied onto the first floppy disk which can be formatted and verified before:

```
fdformat /dev/fd0
dd if=bzImage of=/dev/fd0 bs=512
```

For the second floppy we can find a root fs image on the Linux install CD or somewhere on the web. The content of a root floppy can be modified in a very simple way. Mount the rootfs image file via the loopback device:

```
mount rootfs.img /mnt/aux -o loop
```

In /mnt/aux one will see the content of the root filesystem. One can copy extra files on to it if necessary and if there is enough free space on it. After umounting the image file must be compressed:

```
gzip -c9 rootfs.img > rootfs.img.gz
```

and the size of the compressed file must be within the floppy size. It can be copied on to the second floppy just as the kernel image on to the first one.

Floppy disk drive should be set as the first boot device in CMOS setup. We start with the first (kernel) floppy. When it asks for the second one ("Insert floppy with root fs and press ENTER") change the floppy. After loading the root filesystem the second floppy can be taken out as well.

Remarks

Booting a PC from floppy disk(s) is the absolutely minimal requirement. If we have a CD/DVD in each computer or their BIOS can be boot from pen-drive we are in a more comfortable position.

In my case (configuration is at the top) the filesystem transfer (second method, about 1 GB image size) takes up about 3 (three) minutes which is not enough to boot the following machine and make the corrections of the previous one.

In the second case the hardware configuration of the computers may vary in a wide range. The only criterium is that we must produce a kernel on the source machine which is able to boot on each cloned machine as well. If a cloned machine can boot all the other necessary configuration steps can be achieved in batch mode from a remote computer via ssh.

References

PETRELEY, N. & BACON, J. (2006). *Linux asztali gépen*. Budapest: O'Reilly & Kiskapu.
HAGEN, B. VON & JONES BRIAN, K. (2006). *Linux bevetés közben. Második küldetés*. Budapest: O'Reilly & Kiskapu.