How to clone a Linux box using netcat

Dr. Emiliano Bruni

info@ebruni.it

Version 0.01
$Date: 2003/08/22 14:52:15 $

Many times it’s necessary to have a clone of a linux box without modifying the original box. With a clone box you can have a backup or failover server, testing and developing new software solutions witho no trouble for the original services. The best way that I had found, was to use **dd** to clone the original hard disk to a new one. This solution take a lot of time with current large disks and requires, to work well, to be in single-user mode and to add and then remove the backup hard disk. This document describes an alternative solution of the problem that is more faster, works while the original box is working and it’s no necessary to power off it because copy will be made to another box using netcat.

How to clone a Linux box using netcat

Revision History

Revision 0.01 2003-08-22
First release.

Table of Contents

1. Introduction

1.1. Why another method for cloning a Linux box?

When i tried to find methods to clone a Linux box, i saw that the only good choise to do this was to use **dd** to duplicate an hard disk to a new one.

This tool is a great tool when necessuty was to be sure that the cloned disk is identically to the original one, moreover thios method is indipendent to data present into the disk. Being a bit-to-bit copy of the original disk to the cloned one, it creates exactly the same partition structure and the same bootable flags.

But this method suffers of some not secondary problems. First of all it's necessary to install a second disk to the linux box and so it's necessary to poweroff the computer that it's offline during installation. Even if it's possible to use the same tool we use in this document "netcat" to redirect data **dd** output to another computer, this possibility is negated from the fact that, to be a consistent copy, the original computer must be put in single-user mode to avoid that data changes during copy. In a copy bit-to-bit it's highly probabily to have data corruption if data changes during copy. Also if, in single-user mode, it's possible to make network link up, this is highly discouraged.

Moreover, the fact that **dd** make a bit-to-bit copy, implies that it's necessary to copy empty disk sector too which, related to the increase of the dimension of disks, makes that it's necessary a lot of time to copy a disk to another.

Owing to these problems this method it's not a good choise when the source computer it's a production online server.

1.2. Is there a solution?

To find a satisfactory solution to the problem of backup an whole disk into another i try to find some tools that solve problems seen over but with same advantages of **dd**.

Not having to use **dd** i try to use the other achive tool of unix family: **tar**. Locally **tar** program works well to produce an archive of whatever directory and, in union with a compressing tool like **gzip** or **bzip** is the standard way to distibute package programs.

Now that I have found a good archive tool, I need a system to not install the destination disk into the source computer. Solution could be to install it to another computer and using network to transfer data to be copied. The problem is that the destination computer could have this only hard disk and so not to have an operating system and complex client-server software to support network operation.

We'll see that the union of netcat tool with a CD linux distribution can solve the problem.

2. The example scenario.

To explain how to clone a linux box into another we try to use a supposed scenario with two computer: a source computer and a destination computer where we try to duplicate the first one.

To complicate the scenario we supposed that the source Linux box is installed together with another system operating while the destination box is configured with an empty hard disk that we wish to complete dedicate to backup the only Linux box so the source and destination partition table will be different. Moreover we required not to shutdown or make offline the source computer so that it continues to offer its remote services.

2.1. The source computer

Source computer have one IDE hard disk and a network card. It's a RedHat distribution with grub boot loader and is running a lot of network services.

Linux is installed on `/dev/hda6` partition while `/dev/hda7` is used like a swap linux partition. The boot loader have another menu' item to bootstrap a different operating system installed on `/dev/hda1`.

2.2. The destination computer

Destination computer have one empty IDE hard disk and a network card.

3. Preparing the destination computer

3.1. How to run netcat on destination computer?

To let the destination computer to be visible on the network, to run netcat and to mount the destination hard disk, an operating system is required to be running on this computer.

Where lives this OS if its hard disk is empty. We could install another hard disk, to use the first one for installing a Linux distribution with netcat support and to use the other one clone the original box. We could prepare a linux bootable floppy that create a root ram disk that can enable networking and with a static linked version of netcat tools. But the best and fast way i found is to use a CD bootable Linux distribution like knoppix (www.knoppix.de) which have all we need (and also over) for our scopes. In the next of the document we referer to Knoppix Live CD version 3.2.

So, let's download (???) this CD Live distribution and let's copy it to a writable CD-ROM. Be sure that the computer bios boot sequence begins with CD-ROM drive, insert Knoppix in the CD drive and power on this computer.

After boot sequence we have a running Linux box that only use the CD drive. We have now an environment where bigins our path to backup the source box by partitioning the destination disk.

3.2. Partitionate the destionation disk.

First of all we need to partitionate the destination disk. Since original filesystem it's located in a single partition plus a swap partition we must create two partition on this new disk.

Open a shell console terminal, using **su -** become root and start **fdisk**.

```
knoppix@0[knoppix]$ su -
root@0[root]# fdisk /dev/hda
```

Type **p** on the **fdisk** command line to see disk information and to be sure that it's empty.

```
Command (m for help): p

Disk /dev/hda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
```

and, as you can see, destination hard disk is not partitionating and is 4.3 Gbyte.

Let's go to create a new swap partition with side double as the available RAM memory. In our scenario RAM size is 256 Mbyte so we creare a swap partition of 512 Mbyte:

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-522, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-522, default 522): +512M

Command (m for help): p

Disk /dev/hda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1        63    506016   83  Linux
```

This partition is now a standard linux partition (`Id=83`). Let's modify `Id` type to `82` to change it to a Linux swap partition.

```
Command (m for help): t
Selected partition 1
Hex code (type L to list codes): 82
Changed system type of partition 1 to 82 (Linux swap)

Command (m for help): p

Disk /dev/hda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1        63    506016   82  Linux swap
```

Let's create the Linux system partition

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (64-522, default 64):
Using default value 64
Last cylinder or +size or +sizeM or +sizeK (64-522, default 522):
Using default value 522

Command (m for help): p

Disk /dev/hda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1        63    506016   82  Linux swap
/dev/hda2            64       522   3686917+  83  Linux
```

set the bootable flag to `/dev/hda2`

```
Command (m for help): a
Partition number (1-4): 2

Command (m for help): p

Disk /dev/hda: 4294 MB, 4294967296 bytes
```

```
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1        63    506016   82  Linux swap
/dev/hda2   *        64       522   3686917+  83  Linux
```

and commit changes to partition table:

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

The destination disk is now properly partitionated. We proceed to format it.

3.3. Format the destination disk

To use swap partition and to mount linux partition we need to format them. First of all we can format the swap partition and using it to make faster next operations.

```
root@0[root]# mkswap /dev/hda1
Setting up swapspace version 1, size = 518156 kB
root@0[root]# swapon /dev/hda1
```

Then we can format the linux partition using `-j` flag to format it with ext3 journaled filesystem

```
root@0[root]# mke2fs -j /dev/hda2
mke2fs 1.33 (21-Apr-2003)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
461216 inodes, 921729 blocks
46086 blocks (5.00%) reserved for the super user
First data block=0
29 block groups
32768 blocks per group, 32768 fragments per group
15904 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Now that `/dev/hda2` is an ext3 linux filesystem we can mount it. Create a directory `/mnt/hda2` on Knoppix ramdisk filesystem and mount the real hard disk on it:

```
root@0[root]# mkdir /mnt/hda2
root@0[root]# mount /dev/hda2 /mnt/hda2/
root@0[root]# ls -al /mnt/hda2/
total 21
drwxr-xr-x    3 root     root         4096 Aug 13 10:06 .
drwxr-xr-x    7 root     root         1024 Aug 13 10:14 ..
drwx------    2 root     root        16384 Aug 13 10:06 lost+found
```

Now we can complete the destination computer configuration enabling networking.

3.4. Enable networking on destination computer

Our scope is to transfer original box to destination box over a network connection so, supposing the original computer already has a working network configuration, we need to enable network on the destination computer.

A lot often the transfer will happen between two computer on the same local net therefore the destination computer will have to be configured with an address pertaining to the same range of the compute sorce.

Supposing that the original box have ip address as 192.168.0.1 with a netmask 255.255.255.0 or, in other terms, a /24 network range we can configure the destination box with a free ip address in the same range, say 192.168.0.2, from a terminal console with this commands

```
root@0[root]# ip link set eth0 up
root@0[root]# ip address add 192.168.0.2/24 dev eth0
```

After this it could be possible to ping the original box

```
root@0[root]# ping -c 1 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: icmp_seq=0 ttl=255 time=6.4 ms

--- 192.168.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 6.4/6.4/6.4 ms
```

Now all is ready to begin tranfer data over network. We can now use netcat to clone the original box.

4. Tranfer data over network

To clone the original box to the destination box over network using netcat we are needed to follow two steps. First of all we transfer a partial portion of original disk to permit us to abandon the CD linux distribution and to make possible to boot the destionation box directly from hard disk.

When, on the destination box, it runs a Linux with the same users who exist on the original box it will be possible to tranfer all data with the safe that permissions and owners of these files will remain unchanged during the tranfer.

But before doing this we need to take familiarity with netcat.

4.1. Netcat usage.

Netcat or **nc** as the actual program is named is a simple utility that can work in two ways

- read data from STDIN and output them through network
- read data from network and output them to `STDOUT`

Data are read/wrote from/through network using raw TCP or UDP protocols. From the point of view of the users, in the first case netcat it is behaved like a *client* that send data to a server while, in the second case, it's behaved like a *server* software binding to a particular port that are wainting to receive data from a client.

The variable that decides if netcat is in client or server mode is the option `-l` that, if present in the command line that runs **nc** told it to listen to a particular port that will be defined using the `-p` option. In client mode

nc *`host port`*

creates a TCP (or UDP if `-u` option is used) to the given port on the given target host, in server mode

nc -l -p *`port`*

put netcat in listening mode to the given port.

Now that we know how netcat utility works we can start to partially tranfer original disk and then to boot destination box directly from hard disk.

4.2. Boot destination box from hard disk.

First of all, in the destination box, move current directory to the mounting point of the empty hard disk and put netcat in listening mode to a randomly port, say `6060`. Since we use tar/gz compressed archive to archive source data we redirect the output that netcat receives to **tar**.

```
root@0[root]# cd /mnt/hda2
root@0[hda2]# nc -l -p 6060 | tar zxvPp
```

We analyze in detail this last line. Like saying, the `-l` option says netcat to put it in listening mode binding it to port `6060` like says by the `-p` option. The pipe "`|`" redirect the `STDOUT` to next command. The `p` flag of the **tar** command says it to read archive data from `STDIN` and then, owing to the pipe, from `STDOUT` of netcat. The other **tar** options should be well known options; `z` says to uncompress data using `gunzip` before trying to explose archive (`x` flag) and `v` is to be verbose. This command continues indefinitely, until the network side of the connection shuts down.

We are now ready to magically clone a directory of the source linux box into the destination box using netcat over network. Let's go to the root directory of the source computer and runs this command

```
  [root@K7 /]# tar czvOPp --same-owner /boot | nc -w 5 192.168.0.2 6060
/boot/
/boot/grub/
...
/boot/vmlinuz-2.4.18-3
[root@K7 /]#
```

If we give a look to the terminal of the destination computer where we left netcat in listening mode we should see something like this:

```
root@0[hda2]# nc -l -p 6060 | tar zxvPp
/boot/
/boot/grub/
...
/boot/vmlinuz-2.4.18-3
root@0[hda2]#
```

and you can see two things, netcat process ends and the verbose messages that are appeared in the source console they are appeared in the destination console too. An **ls** in the destination box confirms us that the source directory is completely backuped to the destination hard disk

```
root@0[hda2]# ls -al boot/
total 4488
drwxr-xr-x    3 root     root         4096 Jul 21  2002 .
```

```
drwxr-xr-x    4 root     root         4096 Aug 13 12:54 ..
lrwxrwxrwx    1 root     root           19 Aug 13 12:54 System.map -> System.map-
2.4.18-3
-rw-r--r--    1 root     root       465966 Apr 18  2002 System.map-2.4.18-
3
...
-rwxr-xr-x    1 root     root      2835238 Apr 18  2002 vmlinux-2.4.18-
3
lrwxrwxrwx    1 root     root           16 Aug 13 12:54 vmlinuz -> vmlinuz-
2.4.18-3
-rw-r--r--    1 root     root      1030147 Apr 18  2002 vmlinuz-2.4.18-
3
```

We more give a deepened glance to the command line we run in the source computer. Let's start from **tar** options:

- `c` indicates that we are creating an archive
- `z` says to compress archive using **gzip**
- `v` is to be verbose
- `O` since we wish to send archive to netcat we don't create an archive file but send archive stream directly to `STDOUT`
- `P` indicates to not remove leading character
- `p` indicates to unchange file permission
- `--same-owner` try to set the same owner to files

so the **tar** create an archive of `/boot` directory compressing it and trying to unchange permission and owner of files. The created archive is send directly to `STDOUT` and is so captured from pipe and redirect on `STDIN` of **nc** command. The **nc** command take inputs from `STDIN` and send it to 192.168.0.2:6060. After `5` seconds from last `EOF` it ends closing local and remote pipe.

We can now tranfer other directory of the original source to create minimun environment to run linux. To do this we need to transfer `/dev` `/lib` `/etc` `/sbin` `/bin` `/usr/sbin` `/usr/bin` `/root` `/var` so execute **nc** in server mode on destination box

```
root@0[hda2]# nc -l -p 6060 | tar zxvPp
```

and copy these directories from source box

```
[root@K7 /]# tar czvOPp --same-owner /dev /lib /etc /sbin /bin \
    /usr/sbin /usr/bin /root /var | nc -w 5 192.168.0.2 6060
```

To run linux we also need to create `/proc` `/initrd` `/mnt` `/tmp` and so

```
[root@K7 /]# mkdir /proc /initrd /mnt /tmp
```

Now we need to edit some files. First of all `/etc/fstab` to change partition device. The more important things are swap and root mounting point that, for how we have partitionated the destination hard disk must link to `/dev/hda1` and `/dev/hda2` and so `/etc/fstab` must be edit like this

```
/dev/hda2               /                       ext3    defaults        1 1
/dev/hda1               swap                    swap    defaults        0 0
none                    /dev/pts                devpts  gid=5,mode=620  0 0
none                    /proc                   proc    defaults        0 0
none                    /dev/shm                tmpfs   defaults        0 0
/dev/cdrom              /mnt/cdrom              iso9660 noauto,owner    0 0
/dev/fd0                /mnt/floppy             auto    noauto,owner    0 0
```

Then we need to change networking information to avoid that at next boot the networking settings of cloned box will be in conflict with the source one. Open `/etc/sysconfig/network-scripts/ifcfg-eth0` and change

```
...
IPADDR=192.168.0.1          (The ip address of the source box)
...
```

to

```
...
IPADDR=192.168.0.2          (The ip address of the destination box)
...
```

Let's change hostname too to avoid confusion. Open `/etc/sysconfig/network` and change

```
...
HOSTNAME=source_host_name          (The hostname of the source box)
...
```

to

```
...
HOSTNAME=dest_host_name          (The hostname of the destination box)
...
```

If network card is a different model it could be necessary to change `/etc/modules.conf`

```
...
alias eth0 pcnet32            (The source network card module
...
```

to another one. Common modules are

- *3c59x* for 3Com EtherLink PCI III/XL cards
- *lance* for AMD LANCE/PCnet cards
- *eepro100* for Intel i82557-559 Ethernet cards
- *ne2k-pci* for NE2000 PCI cards
- *pcnet32* AMD PCnet32 cards

Before restart we need to install bootloader to hard disk boot sector. With GRUB boot loader this is very simple.

First of all take a look at `/boot/grub/grub.conf`. The important thing to look and eventually, to set is the root partition for grub and for kernel. GRUB use a not conventional cwnaming convention to identify hard disk partition. First of all, GRUB requires that the device name is enclosed with '(' and ')', device name is indicated using standard terminology, then there is an integer indicates the drive number and then a second integer indicates the partition number Partition and drive numbers are counted from zero, not from one. These are some examples

- *(fd0)* the first floppy drive
- *(hd0,1)* the second partition of the first drive (`/dev/hda2`)
- *(hd1,0)* the first partition of the second drive (`/dev/hdb1`)

Remember also that GRUB does not distinguish IDE from SCSI. It simply counts the drive numbers from zero, regardless of their type. Knowing this the file `/boot/grub/grub.conf` should look as

```
title Red Hat Linux (2.4.18-3)
    root (hd0,1)          (that is /dev/hda2)
    kernel /boot/vmlinuz-2.4.18-3 ro root=/dev/hda2
    initrd /boot/initrd-2.4.18-3.img
```

To install GRUB into MBR

```
root@0[root]# grub --no-curses

    GRUB  version 0.91  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename. ]

grub> root (hd0,1)
```

```
grub> setup (hd0)
```

We only note that we need to use option `--no-curses` to call **grub** because we still don't have a complete linux box and we can't use curses library support. We can now eject CD and reboot the computer without to be worry of some errors during startup process; we still don't have a complete linux environment. However, afert the startup process, we have a running linux box from HD and we are ready to complete the data transfer.

4.3. Complete cloning

We can now transfer *all* the source box into the destination box. Destination box already have a working network stack so we can simply run again **nc** command in listening mode after to have obtained root privileges and

```
root@0[root]# cd /
root@0[/]# nc -l -p 6060 | tar zxvp
```

On the source computer we can execute a **tar** of the complete archive. But owing to fact that we have modified some source file and owing to fact that `/proc` directory must not be copied we need exclude this items

```
[root@K7 /]# tar czvOPp --same-owner \
    --exclude /proc \
    --exclude /etc/fstab \
    --exclude /etc/sysconfig/network-scripts/ifcfg-eth0 \
    --exclude /etc/sysconfig/network \
    --exclude /etc/modules.conf \
    --exclude /boot/grub/grub.conf \
    / | nc -w 5 192.168.0.2 6060
```

So all source file will be copied into the destination one with same owner and permissions. We can now reboot the destination box and begin to really work on it.