Mandriva Linux: HOWTO Clone a system using KA method

Antoine Ginies

Mandriva Linux: HOWTO Clone a system using KA method
by Antoine Ginies

Published 2007

Revision History

Revision 0.1 OCT 2007 Revised by: ag
update

Table of Contents

..... iv

1. Clone a computer over the network 1
    1.1. KA method 1
    1.2. HOW it works 1
    1.3. Step 1: PXE, TFTP, DHCPD services 1
2. Setup a node as a golden node 5
    2.1. The rescue.sqfs file 5
3. The golden node, KA server 7
4. KA client node 8
    4.1. PXE server (kamethod) 8
    4.2. Stage1 KA method, node waiting stage2 8
    4.3. Stage2, the duplication process 8
    4.4. Prepare the node 8
    4.5. PXE server to local boot 8
5. full log of a KA duplication 9
    5.1. Golden node side 9
    5.2. KA client side 11

Clone a node using KA method

Table of Contents

1. Clone a computer over the network **1**
2. Setup a node as a golden node **5**
3. The golden node, KA server **7**
4. KA client node **8**
5. full log of a KA duplication **9**

1. Clone a computer over the network

Clone a computer over the network

Goal of duplication is to easily deploy a computer over network without taking care of numbers of computer. In this documentation, we call golden node the node we want to clone. We can duplicate SCSI or IDE hard drive, and duplication support multiple filesystem (reiserfs, ext2, ext3, xfs, jfs).

1.1. KA method

KA method

With KA method you can quickly duplicate a node using a *desc* file describing partitions. KA method only duplicate data on partitions, so if you have 80go HDD disk, and only 10go on it, KA only duplicates 10go, and not the whole disk. KA method doesn't not support RAID software.

Drawbacks:

- KA method doesn't support RAID software
- you can only clone Linux filesystems (if you want to duplicate another kinf of FS, it's up to you to modify the scripts)
- you can only duplicate same kind of HDD (IDE or SCSI)

1.2. HOW it works

HOW it works

1.2.1. Steps

Steps

The clone process works in three steps

- *PXE boot to retrieve stage1*: the computer boot on PXE mode, retrieve *vmlinuz* and an *initrd*. The computer is in *stage1* mode, and is able to get the stage2 throug KA. Network is up.
- *get stage2*: the computer gets the stage2 with KA method. The *stage2* contains all necessary tools to recognize your hardware (the most important things is to detect your HDD), and all necessary tools to finalize the cloning process.
- *Duplication process*: the computer auto-probes needed modules to be able to access to HDD. A basic log server is launched on the client node to be able to run command and get status of the KA duplication process.

1.2.2. Needed files

Needed files

All needed files are available on Mandriva Linux cooker.

- *install/stage2/rescue.sqhfs*: this is the stage2 file with all needed files to detect and probe modules, and launch the third step of the duplication process. This file will be used on the golden node.
- *isolinux/alt0/vmlinuz*: linux kernel, needed in the */var/lib/tftpboot/X86PC/linux/images/* directory of the PXE server
- *isolinux/alt0/all.rdz*: stage1 and all needed modules.

1.3.3. PXE configuration

PXE configuration

```
# which interface to use
interface=eth0
default_address=IPADDR_PXE

# the multicast ip address to listen on
multicast_address=224.0.1.2

# mtftp info
mtftp_address=IPADDR_TFTP
mtftp_client_port=1758
mtftp_server_port=1759

# the port to listen on
listen_port=4011

# enable multicast?
use_multicast=1

# enable broadcast?
use_broadcast=0

# user prompt
prompt=Press F8 to view menu ...
prompt_timeout=2

# what services to provide, priority in ordering
# CSA = Client System Architecture
# service=<CSA>,<min layer>,<max layer>,<basename>,<menu entry>
service=X86PC,0,2,linux,Mandrake Linux x86
service=IA64PC,0,2,linux,Mandrake Linux IA64
service=X86PC,0,0,local,Local boot

# tftpd base dir
tftpdbase=/

# domain=guibland.com
domain=
```

1.3.4. DHCPD configuration

DHCPD configuration

IE of an */etc/dhcpd.conf* configuration file. Change *IPADDR_TFTP* with the IP address of the TFTP serrver, and the *NET* value. Don't forget to adjust the *domain-name* and the *domain-name-servers*.

```
ddns-update-style none;
allow booting;
allow bootp;

authoritative;

# Definition of PXE-specific options
# Code 1: Multicast IP address of bootfile
# Code 2: UDP port that client should monitor for MTFTP responses
# Code 3: UDP port that MTFTP servers are using to listen for MTFTP requests
# Code 4: Number of secondes a client must listen for activity before trying
#         to start a new MTFTP transfer
# Code 5: Number of secondes a client must listen before trying to restart
#         a MTFTP transfer
```

```
# define Option for the PXE class
option space PXE;
option PXE.mtftp-ip code 1 = ip-address;
option PXE.mtftp-cport code 2 = unsigned integer 16;
option PXE.mtftp-sport code 3 = unsigned integer 16;
option PXE.mtftp-tmout code 4 = unsigned integer 8;
option PXE.mtftp-delay code 5 = unsigned integer 8;
option PXE.discovery-control code 6 = unsigned integer 8;
option PXE.discovery-mcast-addr code 7 = ip-address;

#Define options for pxelinux
option space pxelinux;
option pxelinux.magic code 208 = string;
option pxelinux.configfile code 209 = text;
option pxelinux.pathprefix code 210 = text;
option pxelinux.reboottime code 211 = unsigned integer 32;
site-option-space "pxelinux";

option pxelinux.magic f1:00:74:7e;
option pxelinux.reboottime 30;

#Class that determine the options for Etherboot 5.x requests
class "Etherboot" {
#if The vendor-class-identifier equal Etherboot-5.0
match if substring (option vendor-class-identifier, 0, 13) = "Etherboot-5.0";
# filename define the file retrieve by the client, there nbgrub
# our tftp is chrooted so is just the path to the file
filename "/etherboot/nbgrub";
#Used by etherboot to detect a valid pxe dhcp server
option vendor-encapsulated-options 3c:09:45:74:68:65:72:62:6f:6f:74:ff;
# Set the  "vendor-class-identifier" field to "PXEClient" in dhcp answer
# if this field is not set the pxe client will ignore the answer !
option vendor-class-identifier "Etherboot-5.0";
vendor-option-space PXE;
option PXE.mtftp-ip 0.0.0.0;
# IP of you TFTP server
next-server IPADDR_TFTP;
}

# create the Class PXE
class "PXE" {
# if the "vendor-class-identifier" is set to "PXEClient" in the client dhcp request
match if substring(option vendor-class-identifier, 0, 9) = "PXEClient";
filename "/X86PC/linux/linux.0";
option vendor-class-identifier "PXEClient";
vendor-option-space PXE;
option PXE.mtftp-ip 0.0.0.0;
next-server IPADDR_TFTP;
}

#host node20 {
#    hardware ethernet 00:40:CA:8C:B6:E9;
#    fixed-address node20;
#}

subnet NET.0 netmask 255.255.255.0 {
  option subnet-mask 255.255.255.0;
  option routers IPADDR_GW;
  default-lease-time 288000;
  max-lease-time 864000;
  option domain-name "guibland.com";
  option domain-name-servers IPADDR_DNS;
  next-server IPADDR_TFTP;
  pool {
  range NET.30 NET.40;
  }
}
```

2. Setup a node as a golden node

Setup a node as a golden node

2.1. The rescue.sqfs file

The rescue.sqfs file

You need the rescue disk (wich contains the */ka* directory), and mount it loop in */mnt/ka*.

```
mkdir /mnt/ka
mount -o loop -t squashfs rescue.sqfs /mnt/ka
```

Go in the */mnt/ka/ka* directory, and see all new files available. All those files are needed to do a *KA* duplication process. We will explain now the rule of each of them. If you want to modify the content of this rescue disk (add scripts...), mount it in a tempory directory, and copy the content into the */mnt/ka* directory.

2.1.1. ka-d.sh

ka-d.sh

This is the master script to declare a node as a golden node. This script takes a lot of arguments.

```
-h, --help : display this message
-n num : specify the number of (destination) nodes
-x dir : exclude directory
-X sdb : exclude sdb for the replication
-m drive : copy the master boot record (for windows) of this drive
-M drive file : use 'file' as master boot record (must be 446 bytes long) for the specified drive
-D partition : also copy partition 'partition'
-p drive pdesc : use 'pdesc' file as partition scheme (see doc) for the specified drive
-d delay : delay beteween the release of 2 clients (1/10 second)
-r 'grub|lilo' : choose the bootloader (you can add mkinitrd options)

ie: ka-d.sh -n 3 -p sda /tmp/desc -X sdb -r 'grub --with=ata_piix --with=piix'
```

2.1.2. replication.conf

replication.conf

This file contain all variables needed by other scripts. It also tries to get information like IP address.

2.1.3. fdisk_to_desc

fdisk_to_desc

This script generate the description table of the hard drive disk in the */tmp/desc* file. This file must follow some rules: one line per partition, with two fields : type of partition and size in megabytes. The type can be linux, swap, extended. Other types can be obtained by appending their hexadecimal number to 'type'. For example linux is the same as type83. The size is either a number of megabytes, or the keyword fill (to take all available space). The logical partitions must have the logical keyword.

2.1.4. gen_modprobe_conf.pl

gen_modprobe_conf.pl

This script create a basic output like the content of the*/etc/modprobe.conf* file. Drawbacks this file must be updated for each new modules available in the kernel (based on the *kernel/list_modules.pm* file).

2.1.5. ka-d-client

ka-d-client

The *ka-d-client* binary file is used to get stage2 with the *KA* method, and after get the whole system. The important argument is the *-s* session name. A *KA* can only connect to a specific session (getstage2, kainstall ...). The code source is available in the ka-deploy-0.92 SRPM.

2.1.6. ka-d-server

ka-d-server

The *ka-d-server* binary file is used to be a *KA* golden node server. Like the *ka-d-client* the session arguments is an important parameter (*-s session_name*). The code source is available in the ka-deploy-0.92 SRPM.

2.1.7. ka_replication.sh

ka_replication.sh

Script launched on the *KA* client (after getting stage2 and probing modules), to do the full process of the *Ka* duplication. This script call other scripts to prepare the node (prepare_node.sh), configure the bootloader (make_initrd_grub or make_initrd_lilo).

2.1.8. store_log.sh

store_log.sh

Basic script to store the log of the *KA* duplication process on an FTP server. Adjust to feet your need, and uncomment the line *#store_log.sh* in the */mnt/ka/ka/ka_replication.sh* file.

2.1.9. bootable_flag.sh

bootable_flag.sh

Script to set bootable an HDD using fdisk. First arg must be the HDD device.

2.1.10. make_initrd_grub

make_initrd_grub

Restore and reload the Grub bootloader in the */mnt/disk* directory. It's a very basic script, and perhaps use the *restore_bootloader* of the Mandriva Linux Rescue should be a better idea.

2.1.11. make_initrd_lilo

make_initrd_lilo

Restore and reload the lilo bootloader in the */mnt/disk* directory. Again it's a very basic script, perhaps we should use the *restore_bootloader* of the Mandriva Linux Rescue.

2.1.12. prepare_node.sh

prepare_node.sh

This script remove in the futur system the old network's udev rules, old dhcp cache files, launch the script *gen_modprobe_conf.pl* to regenerate an uptodate */etc/modprobe.conf* in the new system, and launch the script to restore the bootloader. If you want to do more action on the installed, system, you can modify this script.

2.1.13. send_status.pl

send_status.pl

Very basic perl script to open the port 12345, and paste the content of the */tmp/ka** file. It also permit the execution of commands on node, if user send a message from the golden node with the *exec* prefix.

2.1.14. status_node.pl

status_node.pl

Script to connect to a client node, first arg must be the IP address of the node. You can run command on the node with the *exec* prefix.

3. The golden node, KA server

The golden node, KA server

Now, it is time to build a description of the node partitions. You can use the script */mnt/ka/ka/fdisk_to_desc* as root user, or your favorite text editor, you can write a file like this one:

```
linux 3500
extended fill
logical swap 500
logical linux fill
```

This file describes your partition table and the sample above can be considered as a default one for a recommended installation. There is a 3.5GB / partition, a 500 MB swap partition, and */var* fills the rest, of course you can adjust sizes accoding to your system.

Type the following to start the ka replication server as root user:

```
      <screen>
[root@node40 ka]# ./ka-d.sh -n 1 -p sda /root/desc -X sdb -r "grub --with=jfs --with=ata_piix"
takembr =
desc = sda /root/desc
+ Mount points :
     /dev/sda5 / ext3
     /dev/sda1 swap swap
+ Hard drives :
     sda
+ Reading partition table description for sda
    Added partition 1 : type 82
    Added partition 5 : type 83
+ Included mount points : /
+ Bootloader is: grub --with=jfs --with=ata_piix
+++ Sending Stage2 +++
Compiled : Aug 23 2007 12:58:29
ARGS=+ka-d-server+-s+getstage2+-n+1+-e+(cd /mnt/ka; tar --create --one-file-system --sparse  . )+
Server IP = 10.0.1.40
command = (cd /mnt/ka; tar --create --one-file-system --sparse  . )
I want 1 clients
Socket 4 on port 30765 on node40.guibland.com ready.
Socket 5 on port 30764 on node40.guibland.com ready.
```

- *-r "grub --with=jfs --with=ata_piix"*: use grub bootloader and *--with=jfs --with=piix* mkinitrd option in the chrooted system after the *KA* deploiement
- *-n nb_nodes*: specify how many nodes are clients
- *-p sda/hda desc*: specify if you want to duplicate SCSI or IDE storage, and the name of the hdd
- *-x /tmp*: exclude */tmp* directory
- *-X sdb*: exclude *sdb* hdd for the duplication

Now the node is waiting for the rest of the nodes to start replication.

4. KA client node

KA client node

4.1. PXE server (kamethod)

PXE server (kamethod)

We have to configure the PXE to boot by default on *kamethod*. To do this just edit */var/lib/tftpboot/X86PC/linux/pxelinux.cfg/default* and set *DEFAULT* to kamethod:

```
DEFAULT kamethod
```

So, next time a node boots, the PXE server will force the node to boot using the kamethod.

4.2. Stage1 KA method, node waiting stage2

Stage1 KA method, node waiting stage2

Now, you boot all remaining nodes. The replication process will start once all nodes are up and waiting on the *KA* screen.

If the nodes can't reach the golden node, running the *KA* server the message *Can't reach a valid KA server* will appear. Each node will try five times to reach the *KA* server, after that the node will reboot. As the node boots on *kamethod*, it will retry until it finds it.

4.3. Stage2, the duplication process

Stage2, the duplication process

Once all the nodes have found the *KA* server, the first duplication process will start. This step duplicates the *rescue_stage2* from the */mnt/ka* directory of the golden node, in the client's nodes memory (*/dev/ram3*). Then, nodes chroot their memories (the */tmp/stage2* directory), and launch the *drvinst* command from the rescue disk, to probe all needed their modules (drivers). Then, the second step of the duplication starts.

The duplication process will clone your drives following the description you have made (*/tmp/desc* of the golden node). Nodes will rewrite their partition table, then format their filesystems (ReiserFs, XFS, ext2/3, JFS). All new partitions will be mounted in the */mnt/disk* directory. Then, the drive duplication process will begin. On a fast Ethernet switch you can reach speeds of 10MBytes/sec.

4.4. Prepare the node

Prepare the node

At the end of the duplication process, each node will chroot its partitions and rebuild its */boot/initrd.img*, and */etc/modprobe.conf* files. This step ensures that your node will reboot using its potential SCSI drives and adjusting its network card driver. Before rebooting, each node reinstalls lilo/grub. All your node are now ready, and are clone of master node.

4.5. PXE server to local boot

PXE server to local boot

Don't forget to change the default PXE boot to *local* so node after replication will boot localy.

5. full log of a KA duplication

full log of a KA duplication

5.1. Golden node side

Golden node side

```
[root@node40 ka]# ./ka-d.sh -n 1 -p sda /root/desc -X sdb -r "grub --with=jfs --with=ata_piix"
takembr =
desc = sda /root/desc
+ Mount points :
     /dev/sda5 / ext3
     /dev/sda1 swap swap
+ Hard drives :
     sda
+ Reading partition table description for sda
    Added partition 1 : type 82
    Added partition 5 : type 83
+ Included mount points : /
+ Bootloader is: grub --with=jfs --with=ata_piix
+++ Sending Stage2 +++
Compiled : Aug 23 2007 12:58:29
ARGS=+ka-d-server+-s+getstage2+-n+1+-e+(cd /mnt/ka; tar --create --one-file-system --sparse  . )+
Server IP = 10.0.1.40
command = (cd /mnt/ka; tar --create --one-file-system --sparse  . )
I want 1 clients
Socket 4 on port 30765 on node40.guibland.com ready.
Socket 5 on port 30764 on node40.guibland.com ready.
got UDP packet from 10.0.1.35
Session name matches
Sending UDP reply to 10.0.1.35
Accepting connection from 10.0.1.35
Clients : want_data 0  / connected 0
client says hello !
Client sends options
Client accepts data
Added client 10.0.1.35, daddy = 10.0.1.40
Accepting connection from 10.0.1.35
checking connection auth10.0.1.40 reports 10.0.1.35 has opened data connection
Client 10.0.1.35 reports data position : 0
10.0.1.40 reports 10.0.1.35 has been accepted
Welcome son, you are number 1 (MAX 4)
Let's go!
Total data read = 45 Megs, BUF: 0M  FREE = 34M  startpos = 10M
End of data flow
Dropping children
Dropping child 10.0.1.35
All children dropped
Client says dad disconnected
Client says he has finished
Client has finished transfer
Busy clients: 0 -- connected : 1
Peer closed connection on socket 6
close_connection(6)
Busy clients: 0 -- connected : 0
All clients left, I quit
Total data sent = 48 Megs, in 2172 packets
Transfer time = 6.125 seconds, throughput = 7.881 Mbytes/second
```

```
The pipeline was emptied in 0.026 seconds
 - Sending partition/filesystem/mount points informations...
 +++ Running ka-deploy +++
Compiled : Aug 23 2007 12:58:29
ARGS=+ka-d-server+-s+kainstall1+-n+1+-e+(cd /tmp/ka-d6083 && tar c *)+
Server IP = 10.0.1.40
command = (cd /tmp/ka-d6083 && tar c *)
I want 1 clients
Socket 4 on port 30765 on node40.guibland.com ready.
Socket 5 on port 30764 on node40.guibland.com ready.
got UDP packet from 10.0.1.35
Session name matches
Sending UDP reply to 10.0.1.35
Accepting connection from 10.0.1.35
Clients : want_data 0  / connected 0
client says hello !
Client sends options
Client accepts data
Added client 10.0.1.35, daddy = 10.0.1.40
Accepting connection from 10.0.1.35
checking connection auth10.0.1.40 reports 10.0.1.35 has opened data connection
Client 10.0.1.35 reports data position : 0
10.0.1.40 reports 10.0.1.35 has been accepted
Welcome son, you are number 1 (MAX 4)
Let's go!
Total data read = 0 Megs, BUF: 0M  FREE = 34M  startpos = 0M
End of data flow
Dropping children
Dropping child 10.0.1.35
All children dropped
Client says dad disconnected
Client says he has finished
Client has finished transfer
Busy clients: 0 -- connected : 1
Peer closed connection on socket 6
close_connection(6)
Busy clients: 0 -- connected : 0
All clients left, I quit
Total data sent = 0 Megs, in 1 packets
Transfer time = 0.016 seconds, throughput = 0.628 Mbytes/second
The pipeline was emptied in 0.027 seconds
 WAITING node (partition/format)
 - Sending Linux filesystem...
 +++ Running ka-deploy +++
Compiled : Aug 23 2007 12:58:29
ARGS=+ka-d-server+-s+kainstall2+-n+1+-e+(cd /; tar --create --one-file-system --sparse  )+
Server IP = 10.0.1.40
command = (cd /; tar --create --one-file-system --sparse  )
I want 1 clients
Socket 4 on port 30765 on node40.guibland.com ready.
Socket 5 on port 30764 on node40.guibland.com ready.
got UDP packet from 10.0.1.35
Session name matches
Sending UDP reply to 10.0.1.35
Accepting connection from 10.0.1.35
Clients : want_data 0  / connected 0
client says hello !
Client sends options
Client accepts data
Added client 10.0.1.35, daddy = 10.0.1.40
Accepting connection from 10.0.1.35
checking connection auth10.0.1.40 reports 10.0.1.35 has opened data connection
Client 10.0.1.35 reports data position : 0
10.0.1.40 reports 10.0.1.35 has been accepted
Welcome son, you are number 1 (MAX 4)
Let's go!
Total data read = 621 Megs, BUF: 24M  FREE = 10M  startpos = 586M
End of data flow
Dropping children
Dropping child 10.0.1.35
```

```
All children dropped
Client says dad disconnected
Client says he has finished
Client has finished transfer
Busy clients: 0 -- connected : 1
Peer closed connection on socket 6
close_connection(6)
Busy clients: 0 -- connected : 0
All clients left, I quit
Total data sent = 627 Megs, in 34011 packets
Transfer time = 127.140 seconds, throughput = 4.937 Mbytes/second
The pipeline was emptied in 1.549 seconds
```

5.2. KA client side

KA client side

Just launch */mnt/ka/ka/status_node.pl IPADD* to get log of the KA client.

```
10.0.1.35> ------| Ka |---- Install starting...
10.0.1.35> Current session is -s kainstall1
10.0.1.35>  Receiving partitions information...OK
10.0.1.35>  Cleaning hard drive...
10.0.1.35> ==> /tmp/kacmd <==
10.0.1.35> Starting log server..
10.0.1.35>
10.0.1.35> ==> /tmp/ka_log-10.0.1.35-20071024-10h32 <==
10.0.1.35> OK
10.0.1.35> Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
10.0.1.35> Building a new DOS disklabel with disk identifier 0x59be1427.
10.0.1.35> Changes will remain in memory only, until you decide to write them.
10.0.1.35> After that, of course, the previous content won't be recoverable.
10.0.1.35>
10.0.1.35>
10.0.1.35> The number of cylinders for this disk is set to 1116.
10.0.1.35> There is nothing wrong with that, but this is larger than 1024,
10.0.1.35> and could in certain setups cause problems with:
10.0.1.35> 1) software that runs at boot time (e.g., old versions of LILO)
10.0.1.35> 2) booting and partitioning software from other OSs
10.0.1.35>    (e.g., DOS FDISK, OS/2 FDISK)
10.0.1.35> Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
10.0.1.35>
10.0.1.35> Command (m for help): The partition table has been altered!
10.0.1.35>
10.0.1.35> Calling ioctl() to re-read partition table.
10.0.1.35> Syncing disks.
10.0.1.35>  Writing partition table for sda using fdisk...OK
10.0.1.35>  Formatting /dev/sda5 as ext3...OK
10.0.1.35>  Formatting /dev/sda1 as swap...OK
10.0.1.35> - Mounting /dev/sda5 as /mnt/disk/ ......OK
10.0.1.35> ++++++++++++++++++++++++++
10.0.1.35> rootfs on / type rootfs (rw)
10.0.1.35> /proc on /proc type proc (rw)
10.0.1.35> none on /sys type sysfs (rw)
10.0.1.35> none on /proc/bus/usb type usbfs (rw)
10.0.1.35> /dev/ram3 on /tmp/stage2 type ext2 (rw)
10.0.1.35> /dev/sda5 on /mnt/disk type ext3 (rw,data=ordered)
10.0.1.35> ++++++++++++++++++++++++++
10.0.1.35> Linux copy is about to start
10.0.1.35> Server IP is 10.0.1.40
10.0.1.35> Buffers names :pipe
Total data received = 620 Megs (11.222 Mbytes/sec); BUF :0M End of data flow
10.0.1.35> Flushing buffers
10.0.1.35> Total data received = 627 Megs, in 434960 packets
10.0.1.35> Elapsed time = 128.482 seconds, throughput = 4.886 Mbytes/second
10.0.1.35>  Syncing disks...OK
10.0.1.35> Linux copy done.
```

```
10.0.1.35> Creating excluded directories
10.0.1.35> - bootloader is grub --with=jfs --with=ata_piix (user choice and options)
10.0.1.35> - Removing duplicated dhcp cache
10.0.1.35> - Writing modprobe.conf
10.0.1.35> ********************
10.0.1.35> install scsi_hostadapter /sbin/modprobe aic7xxx; /sbin/modprobe ata_piix; /bin/true
10.0.1.35> alias eth0 eepro100
10.0.1.35> alias eth1 eepro100
10.0.1.35> ********************
10.0.1.35> - Remove ude network rules
10.0.1.35> removed '/mnt/disk/etc/udev/rules.d/61-net_config.rules'
10.0.1.35> - Running mkinitrd
10.0.1.35> Looking for default grub menu
10.0.1.35> - erase old initrd.img link
10.0.1.35> removed '/mnt/disk/boot/initrd.img'
10.0.1.35>  initrd will be : /boot/initrd-2.6.22.9-desktop-1mdv.img
10.0.1.35> running: chroot /mnt/disk /sbin/mkinitrd -v -f --with=jfs /boot/initrd-2.6.22.9-desktop-1mdv.im
10.0.1.35> Looking for deps of module aic7xxx
 10.0.1.35>  scsi_transport_spi scsi_mod
10.0.1.35> Looking for deps of module scsi_transport_spi
 10.0.1.35>  scsi_mod
10.0.1.35> Looking for deps of module scsi_mod
10.0.1.35> skip dups
10.0.1.35> Looking for deps of module sd_mod
 10.0.1.35>  scsi_mod
10.0.1.35> Looking for deps of module scsi_wait_scan
 10.0.1.35>  scsi_mod
10.0.1.35> Looking for deps of module ext3
 10.0.1.35>  jbd
10.0.1.35> Looking for deps of module jbd
10.0.1.35> Looking for deps of module jfs
10.0.1.35> Using modules:  /lib/modules/2.6.22.9-desktop-1mdv/kernel/drivers/scsi/scsi_mod.ko /lib/modules
10.0.1.35> Using /tmp as temporary directory.
10.0.1.35> /sbin/nash -> /tmp/initrd.tG1408/bin/nash
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/drivers/scsi/scsi_mod.ko.gz: 60.6%
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/drivers/scsi/scsi_transport_spi.ko.gz: 63.1%
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/drivers/scsi/aic7xxx/aic7xxx.ko.gz: 61.4%
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/drivers/scsi/sd_mod.ko.gz: 60.2%
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/drivers/scsi/scsi_wait_scan.ko.gz: 77.3%
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/fs/jbd/jbd.ko.gz: 60.5%
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/fs/ext3/ext3.ko.gz: 53.9%
 10.0.1.35> /lib/modules/2.6.22.9-desktop-1mdv/kernel/fs/jfs/jfs.ko.gz: 51.4%
10.0.1.35> Loading module scsi_mod.ko
10.0.1.35> Loading module scsi_transport_spi.ko
10.0.1.35> Loading module aic7xxx.ko
10.0.1.35> Loading module sd_mod.ko
10.0.1.35> Loading module scsi_wait_scan.ko
10.0.1.35> Loading module jbd.ko
10.0.1.35> Loading module ext3.ko
10.0.1.35> Loading module jfs.ko
10.0.1.35> /usr/sbin/resume -> /tmp/initrd.tG1408/bin
10.0.1.35> Contents of RCFILE:
10.0.1.35> #!/bin/nash
10.0.1.35>
10.0.1.35> echo "Loading scsi_mod.ko module"
10.0.1.35> insmod /lib/scsi_mod.ko
10.0.1.35> echo "Loading scsi_transport_spi.ko module"
10.0.1.35> insmod /lib/scsi_transport_spi.ko
10.0.1.35> echo "Loading aic7xxx.ko module"
10.0.1.35> insmod /lib/aic7xxx.ko
10.0.1.35> echo "Loading sd_mod.ko module"
10.0.1.35> insmod /lib/sd_mod.ko
10.0.1.35> echo "Loading scsi_wait_scan.ko module"
10.0.1.35> insmod /lib/scsi_wait_scan.ko
10.0.1.35> echo "Loading jbd.ko module"
10.0.1.35> insmod /lib/jbd.ko
10.0.1.35> echo "Loading ext3.ko module"
10.0.1.35> insmod /lib/ext3.ko
10.0.1.35> echo "Loading jfs.ko module"
10.0.1.35> insmod /lib/jfs.ko
```

```
10.0.1.35> echo Mounting /proc filesystem
10.0.1.35> mount -t proc /proc /proc
10.0.1.35> echo Mounting sysfs
10.0.1.35> mount -t sysfs none /sys
10.0.1.35> echo Creating device files
10.0.1.35> mountdev size=32M,mode=0755
10.0.1.35> mkdevices /dev
10.0.1.35> echo Creating root device
10.0.1.35> mkrootdev /dev/root
10.0.1.35> resume
10.0.1.35> echo 1 > /sys/power/suspend2/do_resume
10.0.1.35> echo 1 > /sys/power/tuxonice/do_resume
10.0.1.35> echo Mounting root filesystem /dev/root with flags relatime
10.0.1.35> mount -o relatime --ro -t ext3 /dev/root /sysroot
10.0.1.35> echo Switching to new root
10.0.1.35> switchroot --movedev /sysroot
10.0.1.35> echo Initrd finished
10.0.1.35>  First drive will be: /dev/sda
10.0.1.35> Installation finished. No error reported.
10.0.1.35> This is the contents of the device map /boot/grub/device.map.
10.0.1.35> Check if this is correct or not. If any of the lines is incorrect,
10.0.1.35> fix it and re-run the script `grub-install'.
10.0.1.35>
10.0.1.35> (hd0) /dev/sda
10.0.1.35> (hd1) /dev/sdb
10.0.1.35> umount: /mnt/disk/dev: not mounted
10.0.1.35>  Umounting /dev/sda5...OK
10.0.1.35> AUTH not understood
10.0.1.35> Local directory now /tmp
exec lsmod
10.0.1.35> <console>
10.0.1.35> <console>
10.0.1.35> <console> exec lsmod
10.0.1.35> Module                  Size  Used by
10.0.1.35> aic7xxx               167992  0
10.0.1.35> scsi_transport_spi     22432  1 aic7xxx
10.0.1.35> ata_piix               12228  0
10.0.1.35> libata                109424  1 ata_piix
10.0.1.35> sr_mod                 15044  0
10.0.1.35> sd_mod                 25888  0
10.0.1.35> scsi_mod              124908  5 aic7xxx,scsi_transport_spi,libata,sr_mod,sd_mod
10.0.1.35> loop                   14212  0
10.0.1.35> jfs                   176708  0
10.0.1.35> xfs                   528088  0
10.0.1.35> reiserfs              247908  0
10.0.1.35> ext3                  118824  0
10.0.1.35> jbd                    50184  1 ext3
10.0.1.35> vfat                   10816  0
10.0.1.35> nls_iso8859_1           4672  0
10.0.1.35> nls_cp437               6304  0
10.0.1.35> fat                    45980  1 vfat
10.0.1.35> isofs                  31452  0
10.0.1.35> piix                    9060  0 [permanent]
10.0.1.35> ide_cd                 35488  0
10.0.1.35> ide_disk               14496  0
10.0.1.35> ide_core               99396  3 piix,ide_cd,ide_disk
10.0.1.35> af_packet              17960  0
10.0.1.35> eepro100               28432  0
10.0.1.35> mii                     5376  1 eepro100
10.0.1.35> usbkbd                  6304  0
10.0.1.35> uhci_hcd               22736  0
10.0.1.35> usbcore               113928  3 usbkbd,uhci_hcd
```