Project no. IST-033576

XtreemOS

Integrated Project
BUILDING AND PROMOTING A LINUX-BASED OPERATING SYSTEM TO SUPPORT VIRTUAL ORGANIZATIONS FOR NEXT GENERATION GRIDS

D4.1.1
Initial LinuxSSI integration and packaging in Debian, Mandriva and RedFlag distribution

Due date of deliverable: 28/02/07
Actual submission date: 12/04/07

Start date of project: June 1st 2006

Type: Documentation

Name of responsible: Antoine Giniès
Editor & editor's address: aginies@mandriva.com
nvigier@mandriva.com
Mandriva
43 rue d'aboukir
75002 Paris

Version 1.1/ Last edited by Nicolas Vigier/ Date 12/04/2007

Version	Date	Authors	Institution	Comments
1.0	23/02/07	Antoine Giniès, Nicolas Vigier	EDGE / Mandriva	first version
1.1	28/03/07	Antoine Giniès, Nicolas Vigier	EDGE / Mandriva	Corrections from Jaka Močnik
1.2	11/04/07	Antoine Giniès, Nicolas Vigier	EDGE / Mandriva	Corrections from Bernd Scheuermann
1.3	12/04/07	Antoine Giniès, Nicolas Vigier	EDGE / Mandriva	Corrections from Sandrine L'Hermitte

Abstract

This document describes the integration of LinuxSSI which is based on Kerrighed. It describes the packaging of Kerrighed. The chroot that we use for building Kerrighed is presented with instructions on how to use it. Then automatic rebuild of the packages is described. Finally instructions on how to setup a PXE server and boot a diskless image using dolly are provided in order to be able to test Kerrighed more easily.

Table of Contents

COMMON BASESYSTEM 5
Chroot description 5
Install and configure the chroot 5
Install the chroot 5
Basic configuration 6
Install new software 9
PACKAGING 11
spec file 11
AUTOMATIC PROCESS, UPDATES, PACKAGING 12
Automatic updates 12
Set up automatic build 13
EXPERIMENTAL TEST SUITE PROCEDURE 14
PXE integration 14
Three steps to boot a node with a diskless image 14
PXE configuration, default configuration file 14
Create the stage2 image, script and default configuration file 16
The Diskless image 19
Get the diskless image with Dolly 20
dolly.cfg configuration file 20
Administrate the Kerrighed nodes 23
NEXT STEPS 25
APPENDICES 26
Appendix A : build_kerrighed.pl 26
Appendix B : Setup PXE and DHCPD 30
Appendix C : prepare_diskless_image 34

Common basesystem

A common base system that can be used in a chroot to build XtreemOS has been created.

This XtreemOS chroot is based on the stable 32bits Mandriva GNU/Linux 2007.0 distribution. To quickly sum up the available software is : gcc-4.1.1, gcc3.3 glibc-2.4. For more information about Mandriva GNU/Linux 2007.0 please refer to our website http://www.mandriva.com/.

Chroot description

The goal of this chroot is to provide the same base sytem to all partners in order to avoid confusion that can be caused by having each partner using differents glibc and compiler versions.

This Mandriva GNU/Linux chroot is only used to build XtreemOS during developement. In the end, XtreemOS must remain independent of any GNU/Linux Distribution.

Install and configure the chroot

Install the chroot

You can obtain the chroot environment at this address :

http://people.mandriva.com/~aginies/xtreemos/xtreemos_chroot_20070201.tar.bz2

The **md5sum** file can be found at this address :

http://people.mandriva.com/~aginies/xtreemos/xtreemos_chroot_20070201.tar.bz2.md5sum

The list of already installed RPM in this chroot is available here:

http://people.mandriva.com/~aginies/xtreemos/xtreemos_chroot/data/list_rpms

Check the md5sum of the tarball, and install the chroot. Login with the **root user**:

```
[root@node34 /opt] md5sum xtreemos_chroot_20070201.tar.bz2
a7f5ab31886d10357d1b670a9fcaab97  xtreemos_chroot_20070201.tar.bz2

[root@node34 /opt] cat xtreemos_chroot_20070201.tar.bz2.md5sum
a7f5ab31886d10357d1b670a9fcaab97  xtreemos_chroot_20070201.tar.bz2

[root@node34 /opt] tar xvfj xtreemos_chroot.DATE.tar.bz2
```

The command above will create a directory called **xtreemos_chroot** in your current directory. Now remount the /dev and /proc filesystems in the chroot using the "bind" option, so that theirs contents is available in both places.

```
mount -o bind /dev path_chroot/dev
```

```
mount -o bind /proc path_chroot/proc
```

Basic configuration

In order to setup your internet access use your current **/etc/resolv.conf** file to be able to contact your Internet Domain Name System.

```
cp /etc/resolv.conf path_chroot/etc/resolv.conf
```

Now start the **SSHD** service to be able to connect to the **XtreemOS chroot** via the network. SSH (Secure Shell) is a program for logging into a remote machine and for executing commands in a remote machine.

```
chroot path_chroot service sshd start
```

This SSHD service will not be running on the standard port but on port 24 in order to avoid a conflic with an SSHD service aldready running in the non-chrooted environment.

You can automatically start the SSHD service with a initscript. Create a new file in **/etc/rc.d/ini.d/** called **xtreemos**, with the following content:

```
#! /bin/sh
#
# chkconfig: 2345 61 25
# description: start an sshd daemon on a chrooted XtreemOS system
# processname: sshd
# pidfile: /var/run/xtreemos.pid
# pidfile: $root/var/run/sshd.pid

# PLEASE adjust
root="/path_chroot"

# Get functions
. /etc/rc.d/init.d/functions

# See how we were called.
case "$1" in
 start)
      echo -n "Starting XtreemOS environment: "
      echo
           if [ -f "/var/lock/subsys/xtreemos" ];then echo "Xtreemos envitonment already
running, exiting"; exit 1; fi
      mount /dev/ -o bind $root/dev
      mount -a
      chroot $root mount -t proc /proc /proc
      chroot $root mount -t devpts none /dev/pts
      cp -f /etc/resolv.conf $root/etc/resolv.conf
```

```
        chroot $root service sshd start
        cp -f $root/var/run/sshd.pid /var/run/xtreemos.pid
        echo
        touch /var/lock/subsys/xtreemos
        ;;
 stop)
        echo -n "Stopping XtreemOS chrooted environment: "
        echo
        chroot $root service sshd stop
        chroot $root umount -a
        chroot $root umount /dev/pts
        chroot $root umount /proc
        umount $root/dev
        echo
        rm -f /var/lock/subsys/xtreemos
        rm -f /var/run/xtreemos.pid
        ;;
 status)
        status xtreemos
        ;;
 restart)
        $0 stop
        $0 start
        ;;
 *)
        echo "Usage: $xtreemos {start|stop|status|restart}"
        exit 1
esac
exit 0
```

This initscript should work under Mandriva, Red-Flag and Red-HAT GNU/Linux distribution. For Debian GNU/Linux or other GNU/Linux distributions, contributions are welcome. **ADJUST** the variable **root="path_chroot"** to fit your configuration. It's the full path to where you extracted xtreemos_chroot_DATE.tar.bz2. Now root should be able to start the service, so set this file (/etc/rc.d/ini.d/xtreemos) executable:

```
chmod 755 /etc/rc.d/ini.d/xtreemos
```

SSH service is set by default on port 24 in this chroot. A user **xtreemos** has been created in this chroot, and the default password his **x**. Default password for the root user is **xtreemos**.

Now start/stop/status the **xtreemos** service with **/etc/rc.d/ini.d/xtreemos** script, or **service xtreemos** command. If you want to connect to this chroot from another computer:

```
ssh -p24 xtreemos@IPADDRESS
```

In order to copy data under this chroot:

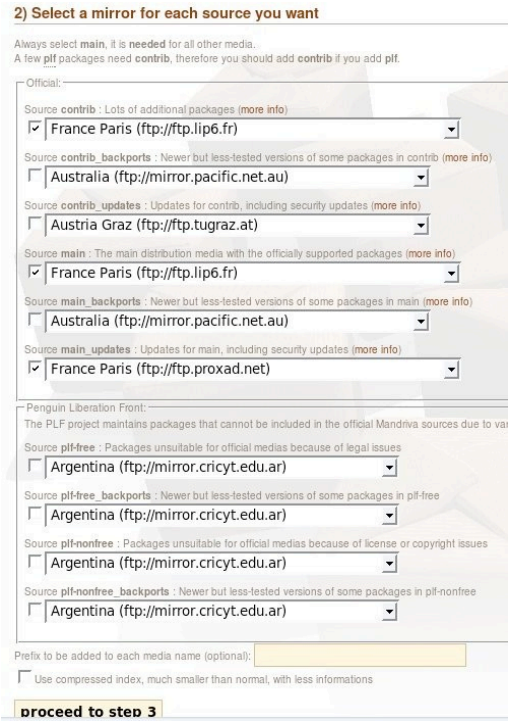
```
scp -P24 data/* xtreemos@IPADDRESS:~/
```

Install new software

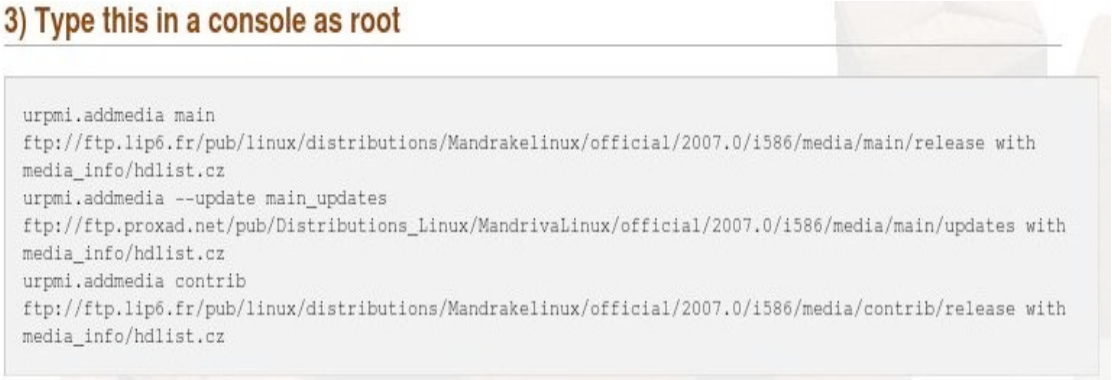
urpmi : the purpose of urpmi is to install rpm packages, including all their dependencies. You need to install a new urpmi database, go to http://easyurpmi.zarb.org/ and follow the steps below :

- choose the Linux distribution, you must choose the 2007.0 version or your chroot will become corrupted

- select a mirror



- add the source repository with the urpmi command line



In order to list all media now available:

```
urpmq --list-media
```

Now you are able to use urpmi. In order to install a package:

```
urpmi name_of_package
```

If you want to find a file in the RPM database, and you don't know in which package it is, so you don't know which package to install, use the:

```
urpmf file
```

or use:

```
urpmf bin/file
```

You can specify a list of packages to skip during installation. Use the file **/etc/urpmi./skip.list**

Packaging

spec file

We use 2 spec files. The one named **kerrighed-kernel-2.6.spec** is used to package the Kerrighed kernel and all modules, and the other one, **kerrighed.spec**, is used to package the libraries and development tools. Those spec files were based on the subversion repository of Kerrighed and are available at :
svn://scm.gforge.inria.fr/svn/kerrighed/packages/rpm/trunk/specs

Specific macro added is:

```
%{?!mdvdis: %define mdvdis %(cat /etc/*release | sort -u | grep -i mandr | wc -l)}
```

This macro returns a value greater than 0 if we use a Mandriva GNU/Linux distribution. There are similar macros for every distributions. Now it's easy to do special things for each distribution with:

```
%if %mdvdis > 0
do something
%else
do something
%endif
```

Automatic process, updates, packaging

Automatic updates

In order to always get up to date packages of Kerrighed to test it more easily as it is evolving, we set up a system to automatically build the packages from the latest sources available in the Kerrighed subversion repository every night. Two computers are involved in this task, the first one is the build server on which we build the packages, and the second one is the FTP server which is accessible from the internet. The packages are built on the first machine inside the chroot that was described earlier in this document.

The script **build_kerrighed.pl** is a perl script which manages the build process, and mirroring to the FTP server which we use for sharing the resulting packages. This script has a few parameters that can be modified :

- The URL of the subversion repository. Default value should be good.
- The path to a local directory where a copy of the sources is stored
- The path to the directory containing the templates for the .spec files that we will use to create the packages. Theses files are available in the subversion repository
- The kernel configuration file that should be used to build the kernel package
- The directory where the resulting packages will be stored before being mirrored to the FTP server
- The maximum number of builds that should be kept. In order to save space, we delete old builds. This value tells us how many different builds should be kept

The main steps of this scripts are the following :

- Get the latest revision number on the subversion repository. This revision number will be used later as a version number in the packages and tarball names
- Update the local source directory to the selected revision number from the subversion repository
- Create the tarball that will be used to build the packages. This is where the kernel patch is generated
- Change the version number inside the .spec files to match the subversion revision we are building
- Build the kerrighed packages. The build logs are kept in a file in the log directory inside the directory which is mirrored to the ftp server. That way, if the build fails, it is possible to see the error messages
- Build the PXE packages
- Move the built packages to the corresponding directory. A directory is created for each revision. Also a symbolic link to that directory is created with the build date as a name. It is then possible to access the packages that have been built from the revision number, or from the build date
- Delete old build directories. In order to save space, we delete them. The number of revisions that should be kept is configurable
- Mirror the files to the ftp server. This is done using rsync

This script can be put in the crontab in order to have a new fresh build every night.

Set up automatic build

In order to start building Kerrighed packages automatically, first create an account on your build server. Let's name it **xtreemos**.

```
adduser xtreemos
su - xtreemos
```

Create the directories required to build RPMS :

```
mkdir -p ~/rpm/{BUILD,RPMS/$ARCH,SOURCES,SRPMS,SPECS,tmp}
```

Replace $ARCH with the architecture for which who want to build your packages.

Now fetch the Kerrighed subversion repository. You only need to get the **trunk** and **packages** directories :

```
mkdir ~/svn_kerrighed
cd ~/svn_kerrighed
svn co svn://scm.gforge.inria.fr/svn/kerrighed/packages
svn co svn://scm.gforge.inria.fr/svn/kerrighed/trunk
```

Create the file ~/.rpmrc and add theses lines :

```
buildarchtranslate:i386:i586
buildarchtranslate:i486:i586
buildarchtranslate:i586:i586
buildarchtranslate:i686:i586
```

Create the file ~/.rpmmacros and add theses lines (change the paths if needed) :

```
%_topdir        /home/xtreemos/rpm
%_tmppath       /home/xtreemos/rpm/tmp
%_packager      XtreemOS <xtreemos@inria.com>
%_signature     gpg
%_gpg_name      XtreemOS Linux
%_gpg_path      ~/.gnupg
%distribution   XtreemOS Linux
%vendor         XtreemOS
```

Now we can run the **build_kerrighed.pl** script, available in Appendix A.

Make sure all the parameters located at the top of the script are correct. This script can be modified and adapted to fit your needs. You can then add this script in the crontab if you want updated packages every night.

Experimental test suite procedure

PXE integration

PXE is a good process to test kerrighed, and especially if we are able to boot a diskless image of the operating system on nodes. This will avoid the long process of installing a node, install the kerrighed kernel, generate the initrd image, reconfigure and re-install the bootloader, and reboot. With a **PXE** and a diskless image, we just have to boot the node with the kerrighed kernel under **PXE**, and send the OS to the node, which will boot it.

An RPM called **pxe_kerrighed-0rVERSION** has been created. It contains the **vmlinuz** and the **all.rdz** image to be able to boot with a PXE (Pre eXecution Environment) server. To create the **all.rdz** image we use the Mandriva's Installer (available on the subversion repository of Mandriva), and patch it to use the kerrighed kernel. We add the 2 network features coming from the Mandriva Clustering product: the capabilities to install the system through **dolly** or **ka** method. Those two installation process are not dependent of the numbers of the nodes, and add a simple way to quickly test kerrighed in diskless mode on a wide range of nodes.
The package **pxe_kerrighed** and the diskless mode to install a node and test kerrighed are still experimental. We encountered some problem because the Mandriva installer use a compressed format called **gzloop**, using a module that is not include in the vanilla kernel.

Three steps to boot a node with a diskless image

1. **PXE boot to retrieve stage1**: the computer boots on **PXE** mode, retrieve **vmlinuz** and an **initrd**. The computer is in **stage1** mode, and is able to get the stage2 through various installation methods (nfs, ftp, http, dolly, ka). Network is up.
2. **get stage2**: the computer gets the stage2 with the method you have chosen. You should use **dolly** or **ka** method to speed up the process of retrieving stage2, and also because kerrighed kernel does not support the **gzloop** image yet. Stage2 contains all necessary tools to recognize the hardware and all necessary tools to be able to run the node in diskless mode.
3. **diskless mode**: the computer automatically probes the required modules, Now you can choose **dolly**, **ka** or **dollyC** to retrieve the OS diskless image. This step can be bypassed, because you can send the full diskless image at step2. We keep step 2 to be able to do a normal installation process.

PXE configuration, default configuration file

To easily test kerrighed we use the **PXE** technology to boot a **kernel**, and an **initrd** image which contains all needed modules for network and media storage. Please, keep in mind that setting up **such services can DISTURB your current network** architecture, so use it with care.

We need a **TFTP** server to share files over the network, in fact the **kernel** and **initrd** image, and a **DHCPD** server which supports **PXE** (various options are needed in the configuration file). You need a **PXE** server to be able to answer **PXE** request from node.

Mandriva GNU/Linux installer supports various methods to install a computer. With **PXE** configuration file you can specify which method you want to use to install your node, or add a specific option at boot prompt. All nodes need to boot with a special **node_id**, so we need to create a **PXE** configuration file for each node. You need to know the MAC address of the node and create a file named : 01-MACADDRR. Edit this file for each node in the directory **/var/lib/tftpboot/X86PC/linux/pxelinux.cfg/**.

For example for the node with the MAC address : 00-02-b3-3f-78-8c, create the file: **/var/lib/tftpboot/X86PC/linux/pxelinux.cfg/01-00-02-b3-3f-78-8c** (don't forget the 01-)

```
PROMPT 1
DEFAULT 2.6.11-krg0r1411_d
DISPLAY messages
TIMEOUT 5

label 2.6.11-krg0r1411_d
MENU LABEL 2.6.11-krg0r1411 auto=dolly
   KERNEL images/vmlinuz-2.6.11-krg0r1411
   APPEND initrd=images/kerrighed-2.6.11-krg0r1411.rdz ramdisk_size=256000 \
automatic=method:dolly,dolly_timeout:100,interface:eth0,network:dhcp rescue  \
dollymethod node_id=1
```

dolly : automatic mode, get stage2 through dolly, set timeout to get stage2 to 100 seconds, if it fails, dolly will try 3 times and reboot. Computer should use eth0 network interface. **rescue dollymethod** words at the end of the line tell the computer to get the system image using the dolly replication method

Create the stage2 image, script and default configuration file

We use the patched rescue disk with **dolly** and **ka** (from Mandriva 2007.0 gi), and we create an image that can be sent with dolly to nodes with a script like the one in Appendix B.

With this script now you can:

- convert a **CLP** image into a ISO9660 image, cause **CLP** is not supported in the vanilla kernel
- create the default configuration file for **PXE**, **tftp**, **DHCPD**

The default **/etc/pxe.conf.stage2**:

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

# the multicast ip address to listen on
multicast_address=224.0.1.2
#multicast_address=192.168.200.1

# 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,XtreemOS Linux x86
service=IA64PC,0,2,linux,XtreemOS Linux IA64
service=X86PC,0,0,local,Local boot

# tftpd base dir
tftpdbase=/
```

```
# domain=guibland.com
domain=
```

The default **/etc/dhcpd.conf.stage2**:

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

not 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 00:50:CA:1C: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;
  }
}
```

```
script to auto-configure PXE and DHCPD server
ie:
./prepare_stage2 -I -p 10.0.1.21 -w 10.0.1.253 -n 10.0.1 -s 10.0.1.253 -t 10.0.1.21

-I: install needed software
-p: IP address of the PXE server (should be this computer)
-w: IP address of the gateway
-n: NET base address for dhcpd conf (ie: 10.0.1)
-s: IP address of the DNS server
-t: IP address of the tftpserver (with vmlinuz and all.rdz)
```

```
ie:
./prepare_stage2 -c path/rescue.clp

You need mount right.
-c: prepare chroot with rescue.clp
```

If you want you can force a node to always get the same IP address. Simply add a dedicated section in **/etc/dhcpd.conf** file:

```
host node31 {
   hardware ethernet 00:02:b3:3f:78:8c;
   fixed-address node31; }
host node32 {
   hardware ethernet 00:02:b3:3f:74:66;
   fixed-address 10.0.1.32; }
```

The Diskless image

With the diskless image it you can test a system which is stored in memory (in a RAM disk in our case). In order to prepare this image, use the script **prepare_diskless_image** from Appendix C.

This script :

- remove the old chroot data (clean_chroot)
- reduce the size of the Linux System image, use a rescue disk (need_rescue)
- install all rpm you want (kerrighed, taktuk2 ...) (install_rpm)
- adjust all configurations for SSH service, needed owner ... (adjust_scripts)
- let you change by hand the configuration before create the diskless image (time_to_change)
- optimize the size on the diskless image (reduce_chroot_size)
- remove all unwanted RPMS (remove_rpm)
- remove the RPM database (clean_rpmdatabase)
- finally create the OS diskless image (create_image)

Create your image with the **kerrighed-kernel.** You need to be the root user:

```
[root@ XTREEMOS chroot experimental_test_suite]$ ls *.rpm
kerrighed-0r1411-1.i586.rpm kerrighed-doc-0r1411-1.i586.rpm
kerrighed-libkerrighed-0r1411-1.i586.rpm  kerrighed-utils-0r1411-1.i586.rpm
kerrighed-devel-0r1411-1.i586.rpm  kerrighed-kernel-2.6.11-krg0r1411.1.i586.rpm
kerrighed-module-0r1411-1.i586.rpm        pxe_kerrighed-0r1411-1.i586.rpm
```

```
[root@ XTREEMOS chroot experimental_test_suite]$ ./prepare_diskless_image create
*.rpm taktuk2
```

The file **diskless_node.img** will be created with all the RPM available in the current directory (all kerrighed package, and the taktuk2 package).

In order to start Kerrighed, you must have one configuration file on each node, the file **/etc/kerrighed_nodes** in the chroot of the diskless_image. This file defines the **session id**, the nodes in the cluster and the network interface used by kerrighed.

```
session=1
nbmin=3
node31:eth0
node32:eth0
node33:eth0
```

Now each node will boot on its PXE configuration file, and will be ready to be part of the Kerrighed cluster.

Get the diskless image with Dolly

Dolly allows you to send a partition or an image to other nodes. We will use it to copy the OS diskless image into the ramdisk of our nodes. To be able to do that, write a configuration file, which describes on which computer you want to send the diskless image, and what kind of diskless image you use (an image, a compressed image, a partition....).

dolly.cfg configuration file

Example of a dolly server configuration file

```
infile diskless_node.img
outfile /dev/ram3
server servernode
firstclient 10.0.1.31
lastclient 10.0.1.33
clients 3
10.0.1.31
10.0.1.32
10.0.1.33
endconfig
```

- **infile diskless_node.img** : input file in the server is diskless_node.img
- **outfile /dev/ram3** : output file on clients. '>' means dolly does not modify the image
- **server servernode** : specify which node is the dolly server
- **firstclient node1** : which node is the first client
- **lastclient node3** : which node is the last one
- **clients 3** : how many nodes
- **10.0.1.31 .. 10.0.1.33**: IP address of the nodes
- **endconfig** : needed, end of configuration file

Choose the **PXE** entry with **automatic=method:dolly** line on the **PXE** server. Prepare your dolly configuration file (see above), and launch:

```
dolly -s -v -f dolly.cfg
```

Now boot all your nodes (here 3 nodes) in **PXE** mode, the OS diskless image will be sent to all of them with the dolly method. Nodes are now ready to test the Kerrighed.

PXE phase:

```
My IP address seems to be 0A000120 10.0.1.32
ip=10.0.1.32:10.0.1.253:10.0.1.253:255.255.255.0
TFTP prefix: X86PC/linux/
Trying to load: pxelinux.cfg/01-00-02-b3-3f-74-66

                Welcome to Mandriva Linux PXE Server
                            Pxelinux



boot:
Loading images/vmlinuz-2.6.11-krg0r1411.....................
Loading images/kerrighed-2.6.11-krg0r1411.rdz.....................
.....................
Ready.
```

Dolly phase:

```
Welcome to XtreemOS ALPHA , Feb 22 2007 15:00:24

 Please wait... .
      .                                                    .
      . Waiting for rescue from Dolly server (timeout 100) (Try 1/3)
```

Using the Diskless image as an operating system:

```
Welcome to XtreemOS ALPHA , Feb 22 2007 15:00:24

proceeding, please wait...
INIT: version 2.86 booting
                    Welcome to XtreemOS Linux
Remounting root filesystem in read-write mode
Mounting proc filesystem
Mounting sysfs on /sys
umount: /stage1/proc/bus/usb: not found
umount: /stage1: device is busy
Loading additional modules...
Installing driver uhci-hcd (for "Intel Corp.|82371AB PIIX4 USB")
usage: modprobe <module> [<options...>]
       failed
Installing driver ata_piix (for "Intel Corp.|82371AB PIIX4 IDE")
insmod: error inserting '/tmp/ata_piix.ko': -1 Unknown symbol in module
insmod ata_piix failed at /sbin/modprobe line 49.
       failed
Installing driver uhci-hcd (for "Intel Corp.|82371AB PIIX4 USB")
usage: modprobe <module> [<options...>]
       failed
Installing driver sonypi (for "Intel Corp.|82371AB PIIX4 ACPI - Bus Master IDE
Controller")
Can't find sonypi.ko in archive
can't find module sonypi
       failed
Installing driver eepro100 (for "Intel Corp.|82559 Fast Ethernet LAN on Motherboard")
Can't find eepro100.ko in archive
can't find module eepro100
       failed
Installing driver eepro100 (for "Intel Corp.|82559 Fast Ethernet LAN on Motherboard")
Can't find eepro100.ko in archive
can't find module eepro100
       failed
Installing driver aic7xxx (for "Adaptec|7892A")
/etc/rc.sysinit: line 67: dhcpcd: command not found
cp: cannot stat `/etc/dhcpc/*': No such file or directory
`/etc/resolv.conf' -> `/tmp/stage2/etc/resolv.conf'
chgrp: invalid group `utmp'
```

```
Stopping sshd:[  OK  ]
Starting sshd:[  OK  ]
running: vol_id /dev/sda1
running: vol_id /dev/sda5
swapon on /dev/sda1
swapon: warning: /dev/sda1 has insecure permissions 0644, 0660 suggested
INIT: Entering runlevel: 3
```

Administrate the Kerrighed nodes

You can use taktuk2 (http://www-id.imag.fr/Logiciels/TakTuk/). The diskless image was configured with an SSH PermitRootLogin yes, and a PermitEmptyPasswords yes. Of course, you need to have installed taktuk2 in the OS diskless image.

```
[a@node34 ~]$ rshp2 -v -l root -m n31 -m n32 -m n33 -c ssh -- uptime
<10.0.1.31> [rank:1]: 14:53:49 up 21 min,  1 user,  load average: 0.02, 0.42, 0.42
<10.0.1.32> [rank:2]: 15:58:59 up 21 min,  1 user,  load average: 0.02, 0.44, 0.43
<10.0.1.33> [rank:3]: 14:58:40 up 21 min,  1 user,  load average: 0.03, 0.47, 0.45

[a@node34 ~]$ rshp2 -v -l root -m n31 -m n32 -m n33 -c ssh -- modprobe kerrighed
```

Log on one node:

```
-- ---
-- KERRIGHED CLUSTER diskless mode ---
-- ---
```

```
[root@10 /]$ kr
krg_capset  krg_join    krg_leave   krg_rsh     krgadm
[root@10 /]$ krgadm
usage:
      krgadm cluster {status|start|stop} [-s] [-n] [-c]
      krgadm nodes {status|add|del|fail|swap|ban|unban} [-s] [-n] [-m]
      krgadm links {status|add|del|fail} [-s] [-l] [-n] [-m]
      krgadm -h this help
Mode:
      cluster clusters management
      nodes   nodes management in a cluster
Options:
      -c number of nodes required
      -l list of link
      -m list of MAC
      -n list of nodes
      -p list of process
      -s subsession id
      -t list of linux thread
[root@10 /]$
```

```
[root@10 /]$ free
             total       used       free     shared    buffers     cached
Mem:        515140     184948     330192          0     160232      12536
-/+ buffers/cache:      12180     502960
Swap:       955828          0     955828

[root@10 /]$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 8
model name      : Pentium III (Coppermine)
stepping        : 6
cpu MHz         : 751.767
cache size      : 256 KB
fdiv_bug        : no
hlt_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 sep mtrr pge mca cmov pat pse36 mmx
fxsr sse
bogomips        : 1486.84
```

Now start the Kerrighed cluster:

```
[root@10 /]$ krgadm cluster star
```

At the end of the **dmesg** command you can see:

```
cluster_nodes: 1 run kerrighed
cluster_nodes: 2 run kerrighed
cluster_nodes: 3 run kerrighed
Kerrighed is running on 3 nodes
```

```
Tasks:  68 total,   1 running,  67 sleeping,   0 stopped,   0 zombie
Cpu0  :  0.3% us,  0.3% sy,  0.0% ni, 99.3% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu1  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu2  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Mem:   1545428k total,   550044k used,   995384k free,   480692k buffers
Swap:  1951816k total,        0k used,  1951816k free,    34424k cached
```

Amount of memory is “1545428”, and we have 3 CPU.

By default the process migration is not allowed for processes. To enable process migration, you have to set capabilities from your shell :

```
[root@10 kerrighed]$ krg_capset -d +CAN_MIGRATE
```

now launch a process like burnBX:

```
[root@10 /]$ burnBX &
[1] 100865
```

the process run on CPU0:

```
Cpu0  : 100.0% us,  0.0% sy,  0.0% ni,  0.0% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu1  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu2  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
```

Migrate the process on another CPU:

```
[root@10 kerrighed]$ migrate 100865 2

Cpu0  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu1  : 100.0% us,  0.0% sy,  0.0% ni,  0.0% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu2  :  0.0% us,  0.0% sy,  0.0% ni, 100.0% id,  0.0% wa,  0.0% hi,  0.0% si
```

Next steps

Include a full process to test the reliability of the Kerrighed kernel and module, and put only tested kernel on the repository.

Appendices

Appendix A : build_kerrighed.pl

```
#! /usr/bin/perl -w

use strict;

use File::Path qw(rmtree mkpath);
use File::Copy qw(move copy);
use POSIX qw(strftime);
use Cwd;

my $SVN_REPOSITORY = 'svn://scm.gforge.inria.fr/svn/kerrighed/trunk';
my $SVN_LOCAL_COPY = $ENV{'HOME'} . '/svn_kerrighed/trunk';
my $TMP_DIR = $ENV{'HOME'} . '/tmp/k';
my $ARCH = 'i586';
my $RPM_DIR = $ENV{'HOME'} . '/rpm';
my $OUTPUT_DIR = $ENV{'HOME'} . '/share/xtreemos';
my $SPECS_TEMPLATES_DIR =
"$ENV{'HOME'}/svn_kerrighed/packages/rpm/trunk/specs";
my $GI_DIR = $ENV{'HOME'} . '/gi/2007.0';
my $KERNEL_VERSION = "2.6.11";
my $KERNEL_CONFIG =
"$ENV{'HOME'}/svn_kerrighed/packages/rpm/trunk/configs/$KERNEL_VERSION/confi
g.$ARCH";
my $ARCHIVE_NUMBER = 7;


my $today = strftime("%Y%m%d_%H%M", localtime());

sub delete_old_builds
{
	my $n = 0;
	chdir("$OUTPUT_DIR/builds");
	foreach (sort glob 'kerrighed-0r*') {
		if ($n > $ARCHIVE_NUMBER) {
			print "Deleting $_ ...\n";
			rmtree($_);
		}
		$n++;
	}
	foreach (glob '*') {
		if (-l $_ && ! -d $_) {
			print "Deleting $_ ...\n";
			unlink($_);
		}
```

```
        }
}

sub make_spec_files
{
        my ($version, $spec_in, $spec_out) = @_;

        open(my $fh_in, '<', $spec_in);
        open(my $fh_out, '>', $spec_out);
        while (<$fh_in>) {
                my $zzz = $_;
                $zzz =~ s/^\%define krgversion .*/\%define krgversion $version/;
                print $fh_out $zzz;
        }
        close($fh_in);
        close($fh_out);
}

sub run_cmd
{
        my $cmd = shift;

        print "+ $cmd\n";
        (system($cmd) == 0) or die "Error running last command";
}

sub make_kerrighed_tarball
{
        my $version = shift;
        #my $kerrighed_dir = "kerrighed-0r$version";

        my $pwd = getcwd;
                        chdir($SVN_LOCAL_COPY) or die "Error entering directory
$SVN_LOCAL_COPY";
        run_cmd("svn -r$version update");
        run_cmd('./autogen.sh');
        run_cmd('./configure');
        run_cmd("make dist-bzip2 PACKAGE=kerrighed VERSION=0r$version");
        move("kerrighed-0r$version.tar.bz2", "$RPM_DIR/SOURCES/");
        chdir($pwd);
}

sub make_pxe_rpm
{
        my $version = shift;
        chdir($GI_DIR);
        foreach (glob 'kernel/RPMS/*.rpm') {
                unlink($_);
        }
```

```
        foreach (glob "$RPM_DIR/RPMS/$ARCH/kerrighed-kernel-$KERNEL_VERSION-
krg0r$version*rpm") {
                copy($_, 'kernel/RPMS/');
        }
        foreach (glob "$RPM_DIR/RPMS/$ARCH/kerrighed-module-0r$version*rpm") {
                copy($_, 'kernel/RPMS/');
        }
                system("./make_pxe_K > $OUTPUT_DIR/logs/$today.build_pxe_kerrighed.txt
2>&1");
}

sub get_latest_version
{
        my $version = `LANG=C svn info $_[0] | sed -e '/^Revision/!d; s/^Revision:[ \t]//;'`;
        chomp $version;
        return $version;
}



my $version = get_latest_version($SVN_REPOSITORY);

print "Latest version is $version\n";

print "Fetching kerrighed sources from svn repository...\n";
make_kerrighed_tarball($version)
        unless -f "$RPM_DIR/SOURCES/kerrighed-0r$version.tar.bz2";
copy($KERNEL_CONFIG,                    "$RPM_DIR/SOURCES/kerrighed-kernel-
$KERNEL_VERSION-krg0r$version-$ARCH.config");

mkpath($OUTPUT_DIR . "/logs");
my $kerrighed_spec = "kerrighed-0r${version}_kerrighed.spec";
make_spec_files("0r$version",           "$SPECS_TEMPLATES_DIR/kerrighed.spec",
"$RPM_DIR/SPECS/$kerrighed_spec");
my $kerrighed_kernel_spec = "kerrighed-0r${version}_kerrighed-kernel-2.6.spec";
make_spec_files("0r$version",   "$SPECS_TEMPLATES_DIR/kerrighed-kernel-2.6.spec",
"$RPM_DIR/SPECS/$kerrighed_kernel_spec");

print "Starting rpm build ...\n";
system("rpmbuild        -ba        $RPM_DIR/SPECS/$kerrighed_kernel_spec        >
$OUTPUT_DIR/logs/$today.build_kerrighed-kernel-2.6.txt 2>&1");
system("rpmbuild            -ba            $RPM_DIR/SPECS/$kerrighed_spec            >
$OUTPUT_DIR/logs/$today.build_kerrighed.txt 2>&1");

make_pxe_rpm($version);

my $output_rpm = "$OUTPUT_DIR/builds/kerrighed-0r${version}/RPMS";
mkpath($output_rpm);
my $output_srpm = "$OUTPUT_DIR/builds/kerrighed-0r${version}/SRPMS";
```

```
mkpath($output_srpm);
symlink "kerrighed-0r${version}", "$OUTPUT_DIR/builds/$today";
unlink("$OUTPUT_DIR/builds/last");
symlink "kerrighed-0r${version}", "$OUTPUT_DIR/builds/last";

foreach (glob "$RPM_DIR/RPMS/$ARCH/*0r$version*.rpm") {
	print "Moving $_ ...\n";
	move($_, $output_rpm . '/');
}

foreach (glob "$RPM_DIR/SRPMS/*0r$version*.rpm") {
	print "Moving $_ ...\n";
	move($_, $output_srpm . '/');
}

delete_old_builds();

system("rsync -avH --delete $OUTPUT_DIR rsync://ftp.ext.mandriva.com/xtreemos");
```

Appendix B : Setup PXE and DHCPD

```
#!/usr/bin/perl -w
#
use strict;
use File::Copy;
use LWP::Simple;
use Getopt::Std;

my $DHCPD_CONF = "/etc/dhcpd.conf";
my $PXE_CONF = "/etc/pxe.conf";
my $PXE_PATH = "/var/lib/tftpboot/X86PC/linux/pxelinux.cfg/";
my $PXE_DEFAULT = $PXE_PATH . "default";
my $PXE_KA = $PXE_PATH . "clone.pxe";
my $mntka = "/mnt/ka";
my $mnttmp = "/tmp/temp_loop";
my $interface = "eth0";

# from MDK::Common, needed because this script will run on non-Mandriva distro
sub cat_ {
  open(my $F, $_[0]) or return; my @l = <$F>; wantarray() ? @l : join '', @l;
}

sub prepare_dhcpd_conf {
  my ($GW, $NET, $TFTP, $DNS) = @_;
  my $FD;
  open($FD, "> /tmp/dhcpd.conf.ready");
  foreach (cat_("/etc/dhcpd.conf.stage2")) {
      s/IPADDR_GW/$GW/g;
      s/NET/$NET/g;
      s/IPADDR_TFTP/$TFTP/g;
      s/IPADDR_DNS/$DNS/g;
      print $FD $_;
    }
  close($FD);
}

sub save_config {
    my ($config) = @_;
    if (!copy($config, $config . ".sav"))
    { print "Can't backup $config: $!\n" }
}

sub prepare_pxe_conf {
  my ($PXE, $TFTP) = @_;
  my $FD;
  open($FD, "> /tmp/pxe.conf.ready");
  foreach (cat_("/etc/pxe.conf.stage2")) {
```

```
    s/IPADDR_TFTP/$TFTP/;
    s/IPADDR_PXE/$PXE/;
    print $FD $_;
 }
 close($FD);
}

sub update_conf {
     print "Do you really want to update your configuration file for PXE and DHCPD
server ? (y/n)\n";
   my $yesno = <STDIN>;
   chomp ($yesno);
   if ($yesno eq "y") {
      if (!copy("/tmp/pxe.conf.ready", $PXE_CONF))
      { print "Can't update PXE configuration: $!\n" }
      if (!copy("/tmp/dhcpd.conf.ready", "/etc/dhcpd.conf"))
      { print "Can't update DHCPD configuration: $!\n" }
      if (!copy($PXE_KA, $PXE_DEFAULT))
      { print "Can't update PXE default configuration files: $!\n" }
   } else {
      print "No update, exiting...\n";
      exit;
   }
}

sub restart_needed_services {
 map { system("service $_ restart"); $_ } qw(dhcpd pxe xinetd);
}

sub save_all_conf {
 map { save_config($_); $_ } $DHCPD_CONF, $PXE_CONF, $PXE_DEFAULT;
}

sub prepare_chroot_img {
 my ($rescue) = @_;
 if (! -e $rescue) { print "Where is the rescue.clp file ?\n"; exit }
 if (-e "/usr/bin/extract_compressed_fs") {
    #    my $FD;
                #            open($FD,   "|extract_compressed_fs   /tmp/patched_rescue.clp   >
/tmp/patched_rescue.iso");
   map { ! -d $_ and mkdir("$_") } $mntka, $mnttmp;
   print "Extract $rescue\n";
   system("extract_compressed_fs $rescue > /tmp/rescue.iso");
   system("umount $mntka");
   print "Mount patched rescue in $mntka\n";
   system("mount -o loop /tmp/rescue.iso $mntka");
   print "Creating stage2.img\n";
   system("dd if=/dev/zero of=stage2.img bs=1M count=45");
   system("/sbin/mkfs.ext2 -q stage2.img");
```

```
    system("mount stage2.img $mnttmp -o loop");
    system("cp -a $mntka/* $mnttmp/; sync");
    system("umount $mnttmp/");
    #system("umount $mntka/");
    print "\n$mntka directory is ready (ka method)\n";
    print "stage2.img is ready (use it in dolly mode: dolly -v -s -f dol)\n";
    print "
Typical dolly.cfg file could be:
------------------------------
infile /tmp/stage2.img
outfile /dev/ram3
server guibpiv.guibland.com
firstclient 10.0.1.34
lastclient 10.0.1.34
clients 1
10.0.1.34
endconfig
------------------------------
";
  } else {
    print "Please install cloop-utils\n";
  }
}

sub install_needed {
  print "- Needed package are: tftp-server dhcp-server pxe\n";
  if (-e "/usr/sbin/urpmi") {
      my $err = eval { system("urpmi --no-verify-rpm --auto tftp-server pxe dhcp-server
cloop-utils") };
#    $err = /1/ and exit;
  } else {
    print "Your are not running a Mandriva system, i can't install needed package: tft-server,
pxe, dhcp-server\n";
  }
}

sub interface_to_ip {
  my ($interface) = @_;
  my ($ip) = `/sbin/ip addr show dev $interface` =~ /^\s*inet\s+(\d+\.\d+\.\d+\.\d+)/m;
  $ip;
}

sub usage {
  print "script to auto-configure PXE and DHCPD server
ie:
$0 -I -p 10.0.1.21 -w 10.0.1.253 -n 10.0.1 -s 10.0.1.253 -t 10.0.1.21

-I: install needed software
-p: IP address of the PXE server (should be this computer)
```

```
-w: IP address of the gateway
-n: NET base address for dhcpd conf (ie: 10.0.1)
-s: IP address of the DNS server
-t: IP address of the tftpserver (with vmlinuz and all.rdz)

ie:
$0 -c path/rescue.clp

You need mount right.
-c: prepare chroot with rescue.clp and a stage2.img
";
 exit;
}

########################################
# MAIN
#
my %opt;
getopts("p:w:n:s:t:c:", \%opt) or usage();
$opt{c} or usage;

if ($opt{c}) {
 prepare_chroot_img($opt{c});
} else {
 #    $opt{pxe} and $opt{gw} and $opt{net} and $opt{dns} and $opt{tftp} or usage;
 save_all_conf;
 # pxe, tftp
 prepare_pxe_conf($opt{p}, $opt{t});
 # GW, net, tftp, dns
 prepare_dhcpd_conf($opt{w}, $opt{n}, $opt{t}, $opt{s});
 update_conf;
 install_needed;
 update_conf;
 restart_needed_services;
}
```

Appendix C : prepare_diskless_image

```
#!/bin/sh
# quick script to setup a cluster chroot
# and create a dolly image
# first arg can be the name of wanted RPMS

# path to the chroot
CHROOT=$HOME/chroot_test
HOSTNAME=`hostname`

# full path to get the rescue image
RESCUE=rescue.clp
# temp ISO
TISO=/tmp/t.iso
# where to mount temp ISO
TLOOP=/tmp/tiso

# image and path to image
img=diskless_node.img
PATH_MOUNT=/tmp/diskless_node

# mandatory RPMS !
MANDATORY="net-tools passwd openssh-server"

mkdir -p $TLOOP $PATH_MOUNT $CHROOT

case $1 in
   createlc)
         shift
         ;;
   *)
         echo " Usage:"
         echo " $0 create package_name package_name2"
         echo
         echo "Auto installed RPMS are:"
         echo $MANDATORY
         exit 1
         ;;
esac

# first arg is list of wanted RPMS
WANTED=$*


clean_chroot() {
   echo " - cleaning old chroot"
   if [ ! $CHROOT ];then exit 1; fi
```

```
    rm -rf $CHROOT/*
}

need_rescue() {
    echo " - needed a basesystem based on rescue"
    if [ ! -f "/usr/bin/extract_compressed_fs" ]; then urpmi cloop ; fi
    extract_compressed_fs $RESCUE > $TISO
    mount -o loop $TISO $TLOOP

    cp -af $TLOOP/* $CHROOT/
    echo " - Copy existing /dev"
    cp -af /dev/* $CHROOT/dev/
    umount $TLOOP
}

install_rpm() {
    echo " - install wanted RPMS"
    echo "mandatory RPMS: $MANDATORY"
    echo "user choice: $WANTED"
        urpmi --auto --ignoresize --split-length 0 --excludedocs --no-verify-rpm --root
$CHROOT $MANDATORY $WANTED
}

adjust_scripts() {
    echo " - special rc.sysinit and go scripts"
    if [ ! -f "rc.sysinit_diskless" ];then echo "can't find rc.sysinit_diskless !" ; exit 1; fi
    cp -avf rc.sysinit_diskless $CHROOT/etc/rc.sysinit
    chmod 755 $CHROOT/etc/rc.sysinit
    cp -avf /etc/profile $CHROOT/etc/profile
    echo "clear" >> $CHROOT/etc/profile
    echo 'export PS1="[\u@\h \W]\$ "' >> $CHROOT/etc/profile
    echo "echo  --                        --- " >> $CHROOT/etc/profile
    echo "echo  -- KERRIGHED CLUSTER diskless mode --- " >> $CHROOT/etc/profile
    echo "echo  --                        --- " >> $CHROOT/etc/profile
    echo "echo" >> $CHROOT/etc/profile

    cat >$CHROOT/ka/go<<EOF
#!/bin/sh
service portmap restart
service sshd restart
EOF

    echo "SWAP=\$(lsparts | grep swap | cut -d ":" -f 1)" >> $CHROOT/ka/go
    echo "swapon -v /dev/\$SWAP" >> $CHROOT/ka/go
# probe the kerrighed kernel
    echo "modprobe kerrighed" >> $CHROOT/ka/go

    chmod 755 $CHROOT/ka/go
    mkdir $CHROOT/root
```

```
    cp -avf /root/.bashrc $CHROOT/root
    mkdir -p $CHROOT/var/lock/subsys/
    touch $CHROOT/etc/redhat-release
# can connect on nodes with empty password
    echo "PermitEmptyPasswords yes" >> $CHROOT/etc/ssh/sshd_config
    perl -pi -e "s/PermitRootLogin.*/PermitRootLogin yes/" $CHROOT/etc/ssh/sshd_config
    chroot $CHROOT /usr/sbin/useradd nobody -d /tmp/nobody -u 1285
    chroot $CHROOT /usr/sbin/groupadd -g 100 all
    chroot $CHROOT pwconv
    chmod 777 $CHROOT/tmp

    mkdir -p $CHROOT/mnt/kerfs/chkpt
    chmod 755 $CHROOT/mnt/kerfs
    echo "none /mnt/kerfs kerfs noauto,defaults,user 0 0" >> $CHROOT/etc/fstab

    cat >$CHROOT/etc/kerrighed_nodes<<EOF
session=1
nbmin=3
node31:eth0
node32:eth0
node33:eth0
EOF
}

remove_X() {
      rm -rf $CHROOT/usr/X11R6
      rm -rf $CHROOT/etc/X11
}

reduce_chroot_size() {
    echo " - try to reduce the size of the chroot"
    rm -rf $CHROOT/usr/share/doc
    rm -rf $CHROOT/boot/*
    rm -rf $CHROOT/core*
    rm -rf $CHROOT/lib/grub
    rm -rf $CHROOT/usr/share/man
    rm -rf $CHROOT/usr/share/mdk/backgrounds
    rm -rf $CHROOT/usr/share/gtk-doc/html/pango/*.png
    rm -rf $CHROOT/usr/share/gtk-doc/html
    rm -rf $CHROOT/usr/lib/perl5/5.8.8/unicore/*.txt
    remove_X
}

remove_rpm() {
    echo " - remove unwanted RPM"
    chroot $CHROOT rpm -e --nodeps -a \
          desktop-common-data mandriva-theme-screensaver \
          qiv mkinitrd libpango1.0_0 libcairo2-1.0.0 libatk1.0_0 libgpm1 \
```

```
        rpm-mandriva-setup popt-data chkconfig cloop-utils smartmontools mandriva-
theme \
        bootsplash wireless-tools sound-scripts gcc-cpp aumix-text soundwrapper mdk-
menu-messages \
        clone urpmi fbgrab gexec authd pxe_kerrighed
}

clean_rpmdatabase() {
   echo " - now we can clean the RPM database"
   rm -rf $CHROOT/usr/share/rpm-helper
   rm -rf $CHROOT/usr/lib/rpm
   rm -rf $CHROOT/var/lib/rpm
}

chroot_size() {
   echo " - size of chroot:"
   du -sh $CHROOT
}

time_to_change() {
   echo
   echo
                                                                                echo
"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
   echo " If you want to change something in the chroot, do it now"
   echo $CHROOT
   echo "$CHROOT/ka/go file is a good place todo that"
   echo "Else just press [ENTER]"
                                                                                echo
"++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"
   read
}

create_image() {
   echo " - create image $img"
   SIZE=`du -sh $CHROOT | cut -d "M" -f 1`

   dd if=/dev/zero of=$img count=${SIZE}999 bs=1100

   echo " - format image $img"
   mkfs.ext2 -m 0 -L "cluster diskless" $img

   echo " - mount loop $img"
   mount -o loop $img $PATH_MOUNT

   echo " - copy all data to image"
   cp -a $CHROOT/* $PATH_MOUNT

   echo " - umount $img"
```

```
    umount $PATH_MOUNT

    echo " - size of $img"
    du -h $img

    echo " - if you want to edit your diskless image, just mount it loop:"
    echo "mount -o loop $img $PATH_MOUNT"
}

end() {
    echo " - create dolly conf like this one:"
    echo "----------------------------"
    echo "infile $img
outfile /dev/ram3
server $HOSTNAME
firstclient 10.0.1.31
lastclient 10.0.1.33
clients 3
10.0.1.31
10.0.1.32
10.0.1.33
endconfig"
    echo "----------------------------"

    echo
    echo " ! DONT forget to adjust the ramsize parameter ! (ie: ram_disk=150000)"
    echo
    echo " - now launch dolly to copy diskless image on nodes:"
    echo "----------------------------"
    echo "dolly -v -s -f dolly.cfg"
    echo "----------------------------"
    echo
}


# main
clean_chroot
need_rescue
install_rpm
adjust_scripts
time_to_change
reduce_chroot_size
remove_rpm
clean_rpmdatabase
chroot_size
create_image
end
```