The Rembo Wizard 2.0

Rembo Server 2.0

Installation and Management

How To

by

Petri Mäkijärvi

April 14, 2003

Abstract:
The Rembo Wizard 2.0 is a free plug-in module for the Rembo Toolkit 2.0, a PXE-enabled, Pre-OS platform for the system hard disk management for Windows and Linux PC-computers.
This document explains the installation and management of the Rembo Toolkit server and The Rembo Wizard on Linux and Solaris based servers. Examples and file paths may contain elements of the author's home institute so some adaptation may be required for the reader's organization.

1 Introduction

The Rembo Wizard is a free plug-in module for the Rembo Toolkit.

This document collects together installation instructions and management policies for all the components that make a single computer as a *Rembo server* with following components and features:

- ISC DHCP 3.0 server that has a hard-coded PXE-redirection to the Rembo Toolkit server as a boot server. Rembo is not used as PXE-proxy in this approach
- Rembo Toolkit 2.0 server
- The Rembo Wizard 2.0 plug-in module
- Backup strategy for the Rembo shared file system and configuration files
- Administrator group of users that can manage DHCP and Rembo services on the server without root privileges

The above is for the Sun Solaris and for Linux based systems. This document does not currently deal with the installations on Windows servers, although all the components are known to be available for Windows 2000 Server.

2 ISC DHCP Daemon 3.0 Installation

This HOW-TO is a supplement to the Internet Software Consortium's documentation for the DHCP 3.0 software. The example Rembo Server platform below is RedHat Linux 7.1 server. Configuration procedure has following objectives:

- Use ISC 3.0 with much richer features than ISC 2.0 found from the standard RedHat Linux 7.1 installation
- Allow a non-root user (but with group level restrictions) to edit DHCP configuration file - with version control - and restart the DHCP daemon
- Add a PXE option space to the DHCP configuration file so that a PC computer that is booting from the network with PXE 2.0 (or greater) could be explicitly told to boot from a given Rembo Server.

In the following procedure DHCP server will be installed on the same RedHat 7.1 Linux machine that is already hosting a Rembo Server. Apart few exceptions, the procedure is know to be the same with SuSE 7.2 Linux and with Sun/Solaris 7/8/9 systems. Differences are explained below where applicable.

2.1 Uninstall the existing dhcp2.x version from a SuSE / RedHat Linux systems

Verify that the dhcpd (2.0) is **not** already installed:

```
ls -l /etc/init.d/dhcpd
ls -l /usr/sbin/dhcpd
```

If installed, uninstall from the RPMs (you can use rpm(8) if you like, instead of graphical interface)

```
export DISPLAY=yourworkstation:0
/usr/bin/kpackage
* Find dhcp-package
* Uninstall the server from the RPM
```

2.2 Installation of the ISC's 3.0 DHCP server

In below example, the source code distribution of ISC <revision> DHCP server is placed in the shared NFS-directory /csadmin/common/install/rembo/dist/. (Not at the ESRF? Obtain your copy from DHCP home page).

```
cd /usr/local
tar xvzf /csadmin/common/install/rembo/dhcpd/dhcp-<revision>.tar.gz

- or Solaris: -
gzcat /csadmin/common/install/rembo/dhcpd/dhcp-<revision>.tar.gz | tar xvf -

- no gcc on Solaris? install it first -
gcc
cd /csadmin/common/sw/install
./cswinstall gcc

cd dhcp-<revision>
./configure
make
make install
make clean

touch /var/state/dhcp/dhcpd.leases
- or Solaris: -
touch /etc/dhcpd.leases

- do you want to remove gcc from Solaris?
cd /csadmin/common/sw/install
./cswremove gcc
```

2.3 Make the DCHP server's restart available for non-root users

The System V init() startup script is OS-dependent, with *.sh*-extension. In below example, the example script is placed in the shared NFS-directory /csadmin/common/install/rembo/dhcpd/. An example of a *dhcpd.sh* for RedHat Linux 7.1 server is available in Appendices 8.1.

```
cd /etc/init.d
cp /csadmin/common/install/rembo/dhcpd/initd.<your_os>/dhcpd.sh
```

Compile on the target system a program that allows any user to call the above */etc/init.d/dhcpd.sh* with supervisor's privileges. For example (the example for the *dhcpd.c*, referred below, is in Appendices 8.4),

```
cd /tmp
cc -o dhcpd /csadmin/common/install/rembo/dhcpd/initd/dhcpd.c
mv dhcpd /etc/init.d/dhcpd
chmod +s /etc/init.d/dhcpd
```

Non-root users should be able to edit dhcpd.conf, create a directory for it (In below example, the example configuration file is placed in the shared NFS-directory */csadmin/common/install/rembo/dhcpd*. An example of *dhcpd.conf* is available in Appendices 8.3. In the example, group *comp* is for the installation personnel):

```
mkdir -p /etc/dhcpdir/RCS
chmod -R 775 /etc/dhcpdir
chown -R <you> /etc/dhcpdir
chgrp -R comp /etc/dhcpdir
cp /csadmin/common/install/rembo/dhcpd/dhcpd.conf_example /etc/dhcpdir/dhcpd.conf
chmod 664 /etc/dhcpdir/dhcpd.conf
chown <you> /etc/dhcpdir/dhcpd.conf
chgrp comp /etc/dhcpdir/dhcpd.conf
```

2.4 *Configure and test DHCP services*

On an other window, login as <*you*> and modify the DHCP configuration to correspond your needs.

```
cd /etc/dhcpdir
ci -l dhcpd.conf
........ do some heavy editing on /etc/dhcpdir/dhcpd.conf .......
* add a network definition for all networks in your system *
```

On the root user's window, check that your modifications do not contain errors

```
/usr/sbin/dhcpd -d -cf /etc/dhcpdir/dhcpd.conf 2>&1 | less

*Common Error* "Can't bind to dhcp address, Address already in use"
*Reason* Remboserver is running and is programmed for DHCPProxy:
*Resolve* Add to rembo.conf:
       DisableDHCPProxy
       BootNoMulticastDiscovery
```

(use *Ctrl-C* with to kill the above process if no errors were reported)

Satisfied? Back in the <*you*> window, save your work

```
ci -u dhdpc.conf (and to get it out next time, use "co -l dhcpd.conf")
```

Check that the startup works, again, being <*you*>, say

```
/etc/init.d/dhcpd start
/etc/init.d/dhcpd stop
/etc/init.d/dhcpd debug
- stop with CTRL-C -
/etc/init.d/dhcpd start
/etc/init.d/dhcpd restart
```

2.5 *Install the dhcpd-service in the startup procedures for System V init()*

```
cd /etc/rc.d/rc3.d
- or Solaris:-
cd /etc/rc3.d

ln -s ../init.d/dhcpd.sh S64dhcpd
ln -s ../init.d/dhcpd.sh K64dhcpd

- RedHat Linux only: -
cd /etc/rc.d/rc5.d
ln -s ../init.d/dhcpd.sh S64dhcpd
ln -s ../init.d/dhcpd.sh K64dhcpd
```

3 Rembo Toolkit Server 2.0 Installation

This HOW-TO is an supplement to the Rembo Server's documentation and to the `INSTALL` file in its distribution. The method explained below prepares the installation for

- Easy upgrade to new patch levels of the Rembo Server by separating the migrating files and directories on a different file system
- Allow a non-root user (but with group level restrictions) to edit Rembo's configuration file - with version control - and restart the Rembo Server
- Prepare the user authentication for The Rembo Wizard plug-in module

3.1 Binary installation of a Rembo Toolkit [version]

For example, Rembo Toolkit *[release]* = 2.0.*[version]* = 014. Distribution is downloaded from the Rembo's web shop and placed in the shared NFS-directory */csadmin/common/install/rembo/dist/* in the following example.

```
cd /opt

tar xvzf /csadmin/common/install/rembo/dist/rembo-[release].[version]_linux.tgz

- or -

tar xvf /csadmin/common/install/rembo/dist/rembo-[release].[version]_solaris.tar

mv rembo-[release] rembo-[release].[version]
rm -f rembo
ln -s rembo-[release].[version] rembo
cd rembo
mv rembo.conf rembo.conf_dist
```

3.1.1 Create a directory for the Rembo shared file system repository

Create /opt/rembo/files to point to a free disk space big enough (20GB+) (if the file system is on NFS, use option *nolock*), for example, on a mounting point */rembo*.

Make the repository file system mounting point writable to the administration group *comp*. Create an emplacement for the *TiNA* backup restorations.

```
chmod 775 /rembo
chown [you] /rembo
chgrp comp /rembo
cd /rembo
mkdir files
ln -s /rembo/files /opt/rembo/files
mkdir Restored
```

3.1.2 Install the default configuration file

Still on the (example */rembo*) file system, with the installation group name *comp*, create configuration file (the example rembo.conf is in the Appendices 8.5).

```
cd /rembo
cp /csadmin/common/install/rembo/conf_examples/rembo.conf .
mkdir RCS
chmod 775 RCS
chown [you] RCS
chgrp comp RCS
ci -u rembo.conf
ln -s /rembo/rembo.conf /opt/rembo/rembo.conf
```

3.1.3 Install the license key

Install the license key that you have obtained from the Rembo Shop (on example, the key is stored in the shared NFS-directory */csadmin/common/install/rembo/dist/* and installed on the */rembo* file system)

```
cd /rembo
cp /csadmin/common/install/rembo/dist/rembo.key_[release] .
ln -s /rembo/rembo.key_[release] /opt/rembo/rembo.key
```

3.1.4 Install *netclnt*-script to protect tty-settings

It has been reported that some versions of *netclnt*-utility program actually modify tty-settings of your console. To avoid this, a small script has been written:

```
#!/bin/sh
termsettings=`stty -g`
/opt/rembo/misc/netclnt $@
stty ${termsettings}
```

The above script is placed into the NFS-distribution directory */csadmin/common/install/rembo/misc*. Install the wrapper script on */usr/local/bin* and set its execution permissions for the system administration group.

```
cd /usr/local/bin
cp /csadmin/common/install/rembo/misc/netclnt .
chmod 775 netclnt
chown [you] netclnt
chgrp comp netclnt
```

3.1.5 Initial start-up

Start The Rembo Server on a new file system

```
cd /opt/rembo/rembo
./rembo -d -v 3
```

Wait until the crypto key is generated and leave the Rembo-process running, with debug printing on a console window.

Log in as root on an other window and install Rembo plugin distribution on Rembo file system

```
cd /opt/rembo
vi srvfiles.nc (replace (pass)word "install" by "rembo")
/usr/local/bin/netclnt srvfiles.nc
```

Stop now the foreground running Rembo-sserver with Ctrl-C.

3.1.6 Install the */etc/init.d* start-up script and its suid-wrapper program

Take the Rembo startup script from the *initd.[os-specific]* directory, and install it *on /etc/init.d*, for example (the rembo.sh referred below is in the Appendices 8.6)

```
cp /csadmin/common/install/rembo/initd.rh71/rembo.sh /etc/init.d/rembo.sh
- or -
cp /csadmin/common/install/rembo/initd.solaris8/rembo.sh /etc/init.d/rembo.sh
```

Compile on the target system a program that allows any user to call the above rembo.sh with supervisor's privileges. For example (the rembo.c referred below is in the Appendices 8.7)

```
cd /tmp
cc -o rembo /csadmin/common/install/rembo/initd/rembo.c
mv rembo /etc/init.d/rembo
chmod +s /etc/init.d/rembo
```

Make sure that there is a symbolic link from Rembo's default installation directory to the actual, */opt* based installation:

```
ln -s /opt/rembo /usr/local/rembo
```

It may occur that TiNA (Time Navigator, a product of Atempo) is used as backup daemon later to backup Rembo file system. Check now that TiNA (or other non-root user) can stop/start Rembo. If OK, leave The Rembo Server running on background.

```
su - tina
/etc/init.d/rembo start
/etc/init.d/rembo stop
/etc/init.d/rembo start
exit
```

Make the required links on the System V *init.d* directories so that Rembo will start automatically on the server, for example, again as root (example for RedHat Linux 7.1 and for Solaris8)
Note: For the startup, you will call the actual script */etc/init.d/rembo.sh* and not the root privilege interface program */etc/init.d/rembo*

```
cd /etc/rc.d/rc3.d
- or Solaris:-
cd /etc/rc3.d

ln -s ../init.d/rembo.sh S65rembo
ln -s ../init.d/rembo.sh K65rembo

- RedHat Linux only: -
cd /etc/rc.d/rc5.d
ln -s ../init.d/rembo.sh S65rembo
ln -s ../init.d/rembo.sh K65rembo
```

3.2 Preparation for the Rembo authentication

The Rembo Wizard contains two built-in user names: *root* and *rembo*. The authentication of these users is done on the server on which the Rembo Server runs. Obviously, it is not a good idea to divulge the root password of the server machine to The Rembo Wizard's end users. It is preferable to create a non-login, public user for the authentication purposes only. Using the operating system's system administration tool (linuxconf, YaST, ...) or just by editing /etc/passwd and /etc/group, do the following:

- Create user *rembo*
  - with a password that would be known by all the personnel involved with the installation
  - /bin/false or similar as shell
  - make member of the installation group, such as *comp*, or similar
- Make sure that *root* belongs to the installation group, as well

The authentication for the installation could be as follows in the *rembo.conf*-file of the Rembo Server:

```
AuthLocalDomain remboauth {
UserGroup "comp"
}
...
GROUP test {
       Options unicast
       AuthDomain "remboauth"
       # triumph
```

```
        Host 00:02:b3:1a:5f:16 {
               StartPage "net://global/rembowiz.shtml"
}
...
```

If the server's operating system supports PAM-authentication, you can create a file */etc/pam.d/rembo* with following, example contents for RedHat 7.1 server (you can use LDAP, or other suitable service for your organization):

```
#%PAM-1.0
auth required /lib/security/pam_stack.so service=system-auth
account required /lib/security/pam_stack.so service=system-auth
```

Following example is for the Solaris 8 PAM authentication file */etc/pam.conf*:

```
# Rembo Client Authentication Requests
rembo auth required /usr/lib/security/$ISA/pam_unix.so.1
rembo account required /usr/lib/security/$ISA/pam_unix.so.1
```

3.3 Preparing Rembo Server for e-mail reporting from The Rembo Wizard

The Rembo Wizard has a built-in reporting features that are directing by default to the host's log file (in */usr/local/rembo/logs/<MAC-address>.log*-file). These reports can be directed to an e-mail address by using the Rembo Toolkit Server as an e-mail relay towards the organization's e-mail server. This is declared in the *rembo.conf*-file of the Rembo Server **before** any hosts declarations:

```
TCPTunnel sendmail {
          RemoteHost "mailserv.mycompany.com"
          RemotePort 25
          }
```

In the *autoload*-file in an appropriate level (global, group, host) you would modify or declare the e-mail addresses to use. Note that the *from* address must be different than the address that receives the reports, otherwise a warning message will pop up on the client computer.

```
str ReportEmail = "sysadmin@mycompany.com";
str FromEmail = "rembo@myremboserver.mycompany.com";
```

4 The Rembo Wizard 2.0 Installation

4.1 Automatic Installation

The automatic installation of The Rembo Wizard is available only if you have GNU-make (*gmake*) program available.

In the below example, The Rembo Wizard's latest version is downloaded from the project's distribution repository

http://sourceforge.net/project/showfiles.php?group_id=46998

and installed in shared administration NFS-directory

```
/csadmin/common/install/rembo/rembowiz-2.0.[version]
```

There is also a symbolic link into the directory where the latest version of The Rembo Wizard has been installed

```
cd /csadmin/common/install/rembo
ln –s rembowiz-2.0.[version] rembowiz
```

Make sure that you are working on machine where the target Rembo Toolkit Server has been installed.
Make sure that the Rembo Toolkit server is running.
Move into the distribution directory and execute the installation part of the Makefile with GNU-make:

```
cd /csadmin/common/install/rembo/rembowiz
make install
- or –
gmake install
```

If you do not have GNU-make available in the computer where the Rembo Toolkit server has been installed but you have a Linux-box around that can mount the same NFS-distribution directory, on that machine say:

```
cd /csadmin/common/install/rembo/rembowiz
make install target=<target_server_name>
```

4.2 Manual Installation

Manual installation of The Rembo Wizard consists of copying several files from the distribution tar-ball into the Rembo Toolkit server's file system. Since the number of files and their names may vary from one distribution to an other you are invited to consult the live web page for the installation:

http://rembowiz.sourceforge.net/sysadm/install.htm#manual

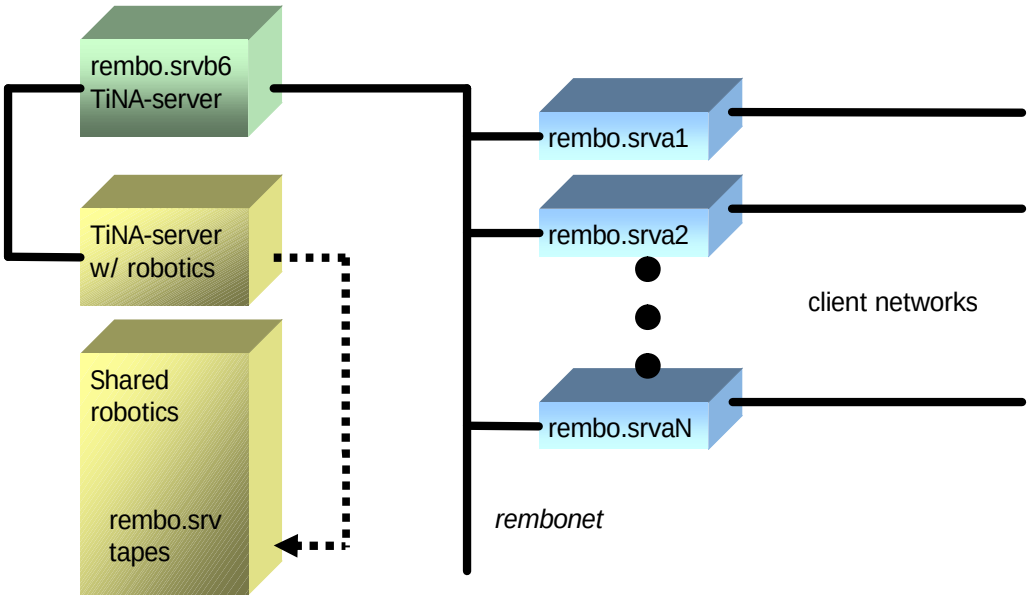
5 Back-up of the Rembo Toolkit server

On a machine installed using the above instructions we can identify easily following file systems that we should back up in regular basis

- */etc/dhcpdir/dhcpd.conf*
  DHCP-declarations of all the machines that should boot from the Rembo server
- */rembo/files*
  Rembo server's shared files repository. This directory contains all the disk images of all the systems that the Rembo server is dealing with. You can consider this file system as a flat file system database that can have any size from 10 MB to 120 GB.
- */rembo/rembo.conf*
  Declarations of all the machines served by the Rembo server

The backup arrangement presented below is from a large scale installation using several Rembo servers. There are several Rembo servers, each serving their own network segment. Each server has a secondary network interface which is connected into an administration network, called *rembonet*. The administration network is used as a channel for the network based backups, using *Time Navigator*, a product of Atempo.

Time Navigator, or *TiNA* server is running on a dedicated backup server. The server does not have a DLT band robotics of its own, but it is getting its slots from an other *TiNA* server that is sharing its robotics over an other network.

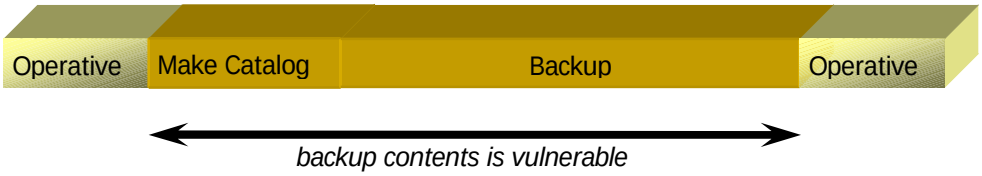


5.1 Running daily backup processes

Starting from Rembo Toolkit Server 2.0 the shared file repository is located on the server's local file system, without any centralized, artificial i-node file that would stay locked. It is therefore possible to take backups of the Rembo Toolkit Server without stopping it first. This is of first importance to the high availability systems, such as control systems and other systems that may require overnight rebooting.

There is a potential pitfall, however that we should be aware of when taking backups.

Let's consider the below diagram. The overnight backup starts and the backup client (in our case *TiNA* client) starts building the catalog of the files it has to backup and then starts to backup them on the backup media.



During the entire backup phase the entire backup contents is vulnerable. Technically speaking we are backing up files but since the Rembo Toolkit stores the client contents on archive files (comparable to the backup catalogs) that points to actual shared files, the backup of *files*-directory can be compared to a backup of a database.

If there will be a client computer image taken during the backup time, the catalog creation or the actual backup can be considered as lost.

Rebooting, restoration or any other client computer activity than the image creation does not have as harmful effect to the backup. The log-files, for example are out of the *files*-directory, on individual files.

5.1.1 Stop Rembo Toolkit Server during the backup process

The advantage of stopping of the Rembo Toolkit Server is, of course that you have more reliable backups. The disadvantage is that you should

- Make sure that the clients will boot even in the absence of the Rembo Toolkit Server.
  - Time-out in PXE boot procedure, next boot device hard disk
  - Bootable hard disk, valid boot sector
    - Windows boot
    - Linux GRUB-boot (LILO is vulnerable, since it contains disk geometry information).

5.1.2 Do not stop Rembo Toolkit Server during the backup process

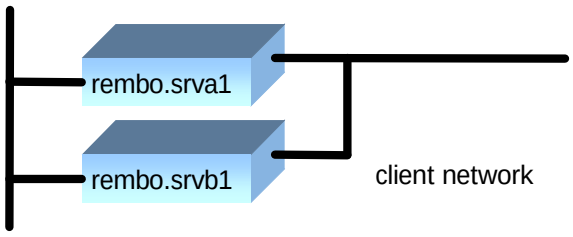
Although the idea of having a possibility for a failed backup may sound bad, in practice the possibility to have one is quite limited:

- Backup is taken during the night and it is not likely that somebody is taking a system image each night at the same time.
  Of course, do not arrange a situation like that by using The Rembo Wizard's *AutoBackup*-feature at the wrong moment.
- Some sub-systems, such as the Control System have a static nature. In such systems it is not allowed to take disk images outside of the maintenance period, for example.
- Before restoring a backup, you can read from Rembo Toolkit's log files if there has been client computer activity during the backup time (unless if the machine is completely lost, of course).

By allowing the Rembo Toolkit Server to run during the backup you will give higher availability of the service without taking excessive risks.

5.1.3 Very High Availability Solution

When none of the solutions explained above is acceptable, there exists a solution that allows very high availability of the service. Both DHCP and Rembo Toolkit Server can be cloned. You can have two servers, *rembo.srva1* and *rembo.srvb1* that can be almost completely identical; apart the server's IP-address. For example, if *rembo.srva1*'s DHCP-server is busy or stopped, *rembo.srvb1*'s DHCP-server will respond, and vice versa.



During the normal operation, secondary DHCP-server on *rembo.srvb1* will point to *rembo.srva1*'s Rembo Toolkit Server. When the backup-time approaches, following will be done:

1. *rembo.srvb1*'s Rembo Toolkit Server is synchronized with *rembo.srva1* using the *netclnt*-utility's *rsync*-command
2. both DHCP-servers are set to point to *rembo.srvb1*'s Rembo Toolkit Server
3. *rembo.srva1*'s Rembo Toolkit Server is stopped
4. backup is taken from *rembo.srva1*
5. *rembo.srva1*'s Rembo Toolkit Server is started
6. both DHCP-servers are set to point back to *rembo.srva1*'s Rembo Toolkit Server

7. reporting messages from The Rembo Wizard are analyzed to see if there is any new images on *rembo.srvb1*
8. if needed, new images from *rembo.srvb1* are copied and installed on *rembosrva1* using the RAD file format (and not *rsync*-command)

5.2 Restoring from the backup

When restoring the Rembo Toolkit Server's shared file system repository in the *files*-directory, two possibilities are envisaged: Restoring a single file or restoring an image archive.

5.2.1 Recovering a single file

There is no need to stop the Rembo Toolkit Server. Using *TiNA*-client, restore the file or files to the original location. Note that only single text files, scripts, plug-in files or such can be restored this way, **not** disk image **archives**.

✍ Keep carefully all the intermediate files in the */rembo/Restored* directory, as illustrated in the below examples. This is to make sure that the *TiNA* backup daemon does not start to make a double backup; the */rembo/Restored* directory is excluded from the backup.

5.2.2 Recovering an image archive

In this procedure you must have **more than 50 per cent disk space** left on the */rembo* file system. Restore the entire *files* directory into the */rembo/Restored* directory, reserved for this purpose. Rename the original shared file system repository directory.

```
/etc/init.d/rembo stop
cd /rembo
mv files Restored/files.org
mv Restored/files .
```

If there is not enough disk space left, please consider to restore the *files*-directory on an other Rembo Toolkit Server. Restart the Rembo Toolkit server on the restored file system.

```
/etc/init.d/rembo start
```

Using either the command line *netclnt*-utility or the Windows-based Server management console, extract the archive and its shared files into a RAD format file
(ex. `radget myimage.bas.rad myimage.bas`).
Before you start, make sure that you have sufficient disk space available. For example, a 1,7 GB Windows XP archive does not fit into a disk space of 1 GB!

Restart the Rembo Toolkit Server with the original shared file system repository

```
/etc/init.d/rembo stop
cd /rembo
mv files Restored/files
mv Restored/files.org files
/etc/init.d/rembo start
```

Using either the command line *netclnt*-utility or the Windows-based Server management console, import the archive and its shared files from the RAD format file.
(ex. `radput myimage.bas.rad myimage.bas`).

Once you have assured that the archive has been restored correctly, remove the restored *files*-directory.

```
cd /rembo
rm –rf Restored/files
```

5.2.3 Recovering the entire shared file system

If we consider that the shared file system repository in the *files*-directory is completely lost we can proceed with **a total replacement** from the backup copy.

```
/etc/init.d/rembo stop
cd /rembo
rm –rf /rembo/files
```

Restore with *TiNA*-client the entire *files*-directory to the original. Check the consistency and take the opportunity to pack the contents of the shared file system using the following commands of the Rembo Toolkit Server. Run the *fsck* equivalent of the Rembo Toolkit, first in reporting only mode:

```
cd /usr/local/rembo
./rembo –d –v 3 –chkshared
```

If there are no **big** warnings, you can run the repair mode on the shared file system

```
./rembo -d –v 3 -fixshared
```

You can get an idea what can be gained if the shared file system is backed by issuing the command

```
./rembo -d –v 3 -statshared
```

Finally, if you consider that the gains are worth of the small risk involved (something **can** go wrong), you can pack the shared file system

```
./rembo -d –v 3 -packshared
```

Start the Rembo Toolkit server.

```
/etc/init.d/rembo start
```

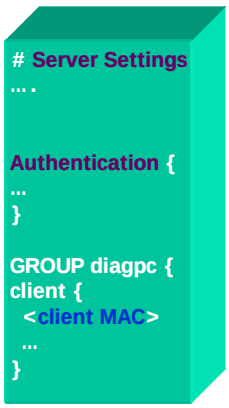
6 Changing Client Computer's MAC-Address

The example platform below is for Solaris 8 server where both DHCP and Rembo Services are running on the same machine. The explained procedure has following objectives:

- Explain how to change host's MAC address on ISC's DHCP 3.0 server's configuration file for
  - PXE option space (Rembo Boot)
  - DHCP option space (Post-Boot Services)
- Explain how to change host's MAC address on a Rembo Server

As an example, host's MAC address changes from *00:02:b3:1a:5f:**16*** to *00:02:b3:1a:5f:**32***.

/etc/dhcpdir/dhcpd.conf



/rembo/rembo.conf

Basically, it is enough to edit two files, */etc/dhcpdir/dhcpd.conf* and */rembo/rembo.conf* and restart the services that use these files. There are three occurrences of a MAC address in the files, as illustrated in the file diagram.

NB: You must do a *telnet, ssh, rlogin* or similar on the target server machine, login as <*you*>. Being *root* and using *su*-command will not work!

Below we log on a Rembo/DHCP server with telnet.

```
telnet <myremboserver>
```

6.1 Modify DHCP Service

Verify that the DHCP configuration file */etc/dhcpdir/dhcpd.conf* is not locked for somebody else (if it is, you must either have to contact the person who forgot the lock on, or to have the super-user privileges to unlock the file)

```
cd /etc/dhcpdir
ls -l
```

Lock the DHCP configuration file for modifications in the *rcs(1)* version control system

```
co -l dhcpd.conf
```

Using *vi, emacs* or what ever editor you prefer, edit the configuration file. Find first occurrence of the old MAC address in the configuration file, from the PXE option space. The following line explains from which Rembo server the system should be boot from. Below the line has been modified for the new MAC address

```
subclass "pxeremboclient_freak" 1:00:02:b3:1a:5f:32; # pckimmo
```

Find now the second occurrence of the old MAC address. It will be in the DHCP option space. Below the MAC address has already been changed to the new value

```
group {
  # Kimmo's PC CS network
  host pckimmo {
        hardware ethernet                       00:02:b3:1a:5f:32;
```

Save the modified *dhcpd.conf* file. Restart the DHCP services with following command to test the modifications

```
/etc/init.d/dhcpd debug
```

If the *dhcpd.conf* file does not contain errors, the DHCP-server will start to run on foreground, sending its output to the console

```
Listening on DLPI/dmfe0/00:03:ba:14:db:37/192.168.1.0/24
Sending on   DLPI/dmfe0/00:03:ba:14:db:37/192.168.1.0/24
Sending on   Socket/fallback/fallback-net
```

If you do not see the above message it is likely that you have a syntax error in the *dhcpd.conf* file. A typical error is a missing semi-colon ";" that you have accidentally deleted. Don't panic, instructions will follow.

If you can see the above text, press *Ctrl-C* to stop the foreground running DHCP daemon.

6.1.1 DHCP Service modified successfully

With no erros in the *dhcpd.file*, continue here. Check-in your work to unlock the *dhcpd.conf* file for future modifications

```
ci -u dhcpd.conf
 <On one line, explain what you have done, give a name of the machine,
 at least>
 <Terminate with a ".">
```

6.1.2 DHCP Service Modifications failed - How to cancel everything

If you cannot figure out what went wrong with the configuration file and why the DHCP service will not start, this is how you can recover the original configuration

```
cp dhcpd.conf /tmp/dhcpd.conf_can_not_understand
rcs -u dhcpd.conf
co dhcpd.conf
   (if asked is it OK to overwrite, answer Y)
/etc/init.d/dhcpd start
ps -ef | grep dhcpd
```

6.2 Modify Rembo Service

Verify that the Rembo Server configuration file *rembo.conf* is not locked for somebody else (if it is, you must either have to contact the person who forgot the lock on, or to have the super-user privileges to unlock the file)

```
cd /rembo
ls -l
```

Lock the Rembo Server configuration file for modifications in the *rcs(1)* version control system

```
co -l rembo.conf
```

Using *vi*, *emacs* or what ever editor you prefer, edit the configuration file. Find the occurrence of the old MAC address in the configuration file. Below the line has been modified for the new MAC address

```
# pckimmo
        Host 00:02:b3:1a:5f:32 {
                StartPage "net://global/rembowiz.shtml"
```

Save the modified *rembo.conf* file. Restart the Rembo Server with following command to make modifications public

```
/etc/init.d/rembo reload
```

The ESRF Rembo Server restart script makes a test to see if the server starts correctly. If you see an error message, it is likely that you have a syntax error in the *rembo.conf* file. Typically this is a missing semi-colon ";" or such.

6.2.1 Rembo Service Modified Successfully

Check-in your work to unlock the *rembo.conf* file

```
ci -u rembo.conf
 <On one line, explain what you have done, give a name of the machine
 at least>
 <Terminate with a ".">
```

In the following step, you would need a network access password to access the Rembo shared file system. It is best viewed with following command

```
cat /rembo/rembo.conf | grep NetPassword
```

6.2.2 Rename MAC-address directory in the Rembo file system

Host's MAC-address should be changed also on the Rembo Server's file system. This is best done with the Rembo Server Management Console. But since it is a Windows program it may not always be available. Following explains how to rename the directory using *netclnt*-tool on the Rembo Server

```
/usr/local/bin/netclnt
Netclnt 2.0 (c) Rembo Technology Sarl
NETFS> connect localhost
Password: <see 6.2.1>
NETFS> cd hosts
Current directory is /hosts
NETFS> move 0002b31a5f16 0002b31a5f32
NETFS> exit
```

7 Scanning Rembo File System With Anti-Virus Software

Prerequisite for the successful scanning of the Rembo shared file system repository is that **no compression is used** when archives are created. Non-compressed archives can be scanned by letting a virus scanning tools, such as InterScan VirusWall.

7.1 Find the host of the origin for an infested shared repository file

When the virus scanning tool reports for an infested file the reported file would be something like:

```
/rembo/files/shared/ABC/DE/blk1234
```

On the above path, *ABC/DE* can be used to find out the IP-address of the infected host. Consider following formula, simplified for the addresses belonging to the same group of class-C IP-addresses):

Host-IP = (Server-IP >> *1)* ^ *ABCDE*

The calculations – although seemingly very simple – can be quite complicated. From the discussions with the Rembo's developers, we can conclude that if the client and server are on the same sub-network, we can divide clients in two address groups, with .1 – .127 and .128 – .255 address ranges, respectively. Few examples:

- Server's IP-address is 192.168.1.8., client's address is 192.168.1.81. What is the *shared/ABC/DE* ?

  ```
  Server IP=.1.8 (hex=0108)
  Client IP=.1.81 (hex=0151)
  (0108 ^ 0151) << 1 = 00B2
  ```

  ABC/DE = 000/B2

- Server's IP-address is 192.168.1.8., client's address is 192.168.1.132. What is the *shared/ABC/DE* ?

  ```
  Server IP=.1.8 (hex=0108)
  Client IP=.1.132 (hex=0184)
  ```

```
(0108 ^ 0184) << 1 = 0118
```
ABC/DE = 001/18

- We have found an infested file in a shared directory *shared/000/06*. The server's IP-address is 192.168.1.8. How to calculate the client's IP-address, supposing that it is on the same sub-network 192.168.1.0 ?

```
Same subnetwork and the ABC=000, use only the lowest byte of the server's
address=08 and DE=06
(06 >> 1) ^ 08 = 0B (11 decimal)
client address is subnetwork+11, 192.168.1.11
```

- We have found an infested file in a shared directory *shared/001/22*. The server's IP-address is 192.168.1.8. How to calculate the client's IP-address, supposing that it is on the same sub-network 192.168.1.0 ?

```
Same subnetwork and the ABC=001, we are on the address range .128 or higher.
(0122 >> 1) ^ 0108 = 0199 (99 hex. = 153 decimal)
client address is subnetwork+153, 192.168.1.153
```

8 Appendices

8.1 /etc etc/init.d/dhcpd.sh used at the ESRF for RedHat 7.1 servers

```
#!/bin/sh
#
# dhcpd         This shell script takes care of starting and stopping
#               dhcpd.
#
# chkconfig: - 65 35
# description: dhcpd provide access to Dynamic Host Control Protocol.

# Source function library.
. /etc/rc.d/init.d/functions

# Source networking configuration.
. /etc/sysconfig/network

# Check that networking is up.
[ ${NETWORKING} = "no" ] && exit 0

[ -f /usr/sbin/dhcpd ] || exit 0
[ -f /etc/dhcpdir/dhcpd.conf ] || exit 0
[ -f /var/state/dhcp/dhcpd.leases ] || exit 0

RETVAL=0
prog="dhcpd"

start() {
       # Start daemons.
       echo -n $"Starting $prog: "
       daemon /usr/sbin/dhcpd -cf /etc/dhcpdir/dhcpd.conf
       RETVAL=$?
       echo
       [ $RETVAL -eq 0 ] && touch /var/lock/subsys/dhcpd
       return $RETVAL
}

stop() {
       # Stop daemons.
       echo -n $"Shutting down $prog: "
       killproc dhcpd
       RETVAL=$?
       echo
       [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/dhcpd
       return $RETVAL
}

# See how we were called.
case "$1" in
  start)
       start
       ;;
  stop)
       stop
       ;;
  restart|reload)
       stop
       start
       RETVAL=$?
       ;;
  condrestart)
```

```
       if [ -f /var/lock/subsys/dhcpd ]; then
           stop
           start
           RETVAL=$?
       fi
       ;;
  status)
       status dhcpd
       RETVAL=$?
       ;;
  *)
       echo $"Usage: $0 {start|stop|restart|condrestart|status}"
       exit 1
esac

exit $RETVAL
```

8.2 /etc etc/init.d/dhcpd.sh used at the ESRF for Solaris servers

```
#!/bin/sh

if [ ! -d /usr/bin ]
then                    # /usr not mounted
        exit 1
fi

#
# Start/stop DHCP service
#


case "$1" in
'start')
        if [ -x /usr/sbin/dhcpd ]
        then
            /usr/sbin/dhcpd -q -cf /etc/dhcpdir/dhcpd.conf > /dev/console 2>&1
        fi
        ;;

'stop')
        oldpid=`/usr/bin/ps -ef |
             /usr/bin/grep '/usr/sbin/dhcpd' |
             /usr/bin/grep -v 'grep' |
             /usr/bin/awk '{print \$2 }'`
        if [ -z "$oldpid" ];then
                echo "No dhcpd process running" > /dev/console 2>&1
        else
                echo "Killing /usr/sbin/dhcpd ($oldpid)" > /dev/console 2>&1
                kill $oldpid
        fi
        ;;
'debug')
        oldpid=`/usr/bin/ps -ef |
             /usr/bin/grep '/usr/sbin/dhcpd' |
             /usr/bin/grep -v 'grep' |
             /usr/bin/awk '{print \$2 }'`
        if [ -z "$oldpid" ];then
                echo "No dhcpd process running"
        else
                echo "Killing currently running /usr/sbin/dhcpd ($oldpid)"
                kill $oldpid
                sleep 2
        fi
        /usr/sbin/dhcpd -d -cf /etc/dhcpdir/dhcpd.conf
```

```
        ;;
'restart')
        oldpid=`/usr/bin/ps -ef |
             /usr/bin/grep '/usr/sbin/dhcpd' |
             /usr/bin/grep -v 'grep' |
             /usr/bin/awk '{print \$2 }'`
        if [ -z "$oldpid" ];then
                echo "No dhcpd process running"
        else
                echo "Killing currently running /usr/sbin/dhcpd ($oldpid)"
                kill $oldpid
                sleep 2
        fi
        /usr/sbin/dhcpd -q -cf /etc/dhcpdir/dhcpd.conf
        newpid=`/usr/bin/ps -ef |
             /usr/bin/grep '/usr/sbin/dhcpd' |
             /usr/bin/grep -v 'grep' |
             /usr/bin/awk '{print \$2 }'`
        if [ -z "$newpid" ];then
                echo "/usr/sbin/dhcpd   **** DID NOT START ****"
                echo "                        *************"
                echo "Do you want me to"
                echo "start it in debug mode? (y/n)"
                read debugit
                if [ "$debugit" = "y" ];then
                        /usr/sbin/dhcpd -d -cf /etc/dhcpdir/dhcpd.conf
                fi
        else
                echo "/usr/sbin/dhcpd STARTED OK (process number $newpid)"
        fi
        ;;
*)
        echo "Usage: /etc/init.d/dhcpd { start | stop | debug | restart }"
        ;;
esac

exit 0
```

8.3 /dhcpdir/dhcpd.conf example with PXE option space pointing to Rembo Boot

```
# dhcpd.conf ESRF DHCP configuration file 3.0
#    Modified: $Date: 2001/11/28 08:53:13 $
# ------------------------------------------------------------------------------
# You may want to have two windows open on this system,
# - one with your own account, one with root priviledges
# 1) The actual location of this file is $Source: /etc/dhcpdir/dhcpd.conf$
#    - move into that directory
# 2) Check out the file for modifications "co -l dhcpd.conf"
# 3) Once modified you may want to stop the dhcpd services
#    "/etc/init.d/dhcp stop
# 4) Check your files against syntax errors by starting dhcpd services with
#    /usr/sbin/dhcpd -d
#    (stop with Ctrl-C)
# 5) Save the modified file, check it in "ci -u dhcpd.conf"
#                                        Please give a meaningful comment.
# 6) run "/etc/initd.d/dhcpd restart"
#    (suid executable that run /etc/initd./dhcpd.sh restart)
# 7) At the end of script: Verify that the dhcpd process actually do start
# ------------------------------------------------------------------------------
# ============================== PXE OPTION SPACE ================================
# Definition of PXE-specific options where the services required by
# the PXE client are answered by the DHCP proxy of the PXE server.
# Rembo server, for example may be located on a different machine than this
# DHCP server. The client is as per Intel's PXE 2.1 specification and the
```

```
# byte ordering is for the Intel processor architecture.

option space PXE;

# Option descriptions and the default values for the ESRF will follow:
# ------------------------------------------------------------------------------
# multicast ip address    : multicast address of boot file
#
# size 4
option PXE.mtftp_ip code 1 = ip-address;

# ------------------------------------------------------------------------------
# discovery control    : find bootserver using .. multicast = 1
#                                                 broadcast = 2
#                                                 use list  = 7
option PXE.discovery_control code 6 = unsigned integer 8;

# ------------------------------------------------------------------------------
# discovery multicast address : Multicast capable Boot Servers listening on this
#
option PXE.discovery_mcast_addr code 7 = array of unsigned integer 8;

# ------------------------------------------------------------------------------
# boot servers         : List of Boot Servers
#                      { type MSB, type LSB, IPcnt, IP-addr-list,
#                        type MSB ... }
#                         - if more boot servers to be listed, redefine
#                        the structure  by adding type MSB,
#                        type LSB and so on.
option PXE.boot_servers code 8 = array of unsigned integer 8;

# ------------------------------------------------------------------------------
# boot menu            : Boot possibilities:
#                      { type MSB, type LSB, desclen, "text", type MSB ... }
#                         - if more items to be listed, redefine the structure
#                        by adding type MSB, type LSB and so on.
#                      - to find out the decimal text format, say ex.:
#                        echo MyMenuText | od -t u1
#
option PXE.boot_menu code 9 = array of unsigned integer 8;

# ------------------------------------------------------------------------------
# menu prompt        : What to display: { timeout, "prompt" }
#
option PXE.menu_prompt code 10 = { unsigned integer 8, text };

# ---------------------- PXE REMBO BOOT CLIENT CLASSES -------------------------
# Create a generic class for PXE clients who wants to boot from the ESRF
# Rembo server(s):
# - Use unicast to connect to the server
# - Boot Server type = 15
# - Rembo Server #1 = example.myprivate.domain (192.168.1.1)
# - Boot menu is currently:
#   (1) "Rembo Server" (default, type 15)
#   (2) "PXE/BootP Server" (if not defined in boot_servers, then local, type 0)

class "pxeremboclient_example" {
       match hardware;
       option vendor-class-identifier     "PXEClient";
       vendor-option-space PXE;
       option PXE.discovery_control 7;
       option PXE.discovery_mcast_addr 0,0,0,0;
       option PXE.boot_servers 0,15,1,10,0,0,1;
       option PXE.boot_menu
```

```
         0,15,12,
         82,101,109,98,111,32,83,101,114,118,101,114,
         0,0,9,
         80,88,69,47,66,111,111,116,80;
       option PXE.menu_prompt 3
             "EXAMPLE responded: Boot from a Rembo Server: EXAMPLE (F8=menu)";
}
# Following is the list of all Rembo clients, selected by the first broadcast
# packet's hardware information fields.
# - The format is <ARP-hardware type><hardware address>, for Ethernet
#   1:<MAC-address>
#
subclass "pxeremboclient_example" 1:00:02:b3:1a:5f:16; # node1

# ============================== DHCP OPTION SPACE ==============================

# option definitions common to all hosts
#
server-identifier               example.myprivate.domain;
server-name                     "example";
option domain-name              "myprivate.domain";
option domain-name-servers      192.168.1.1;
option subnet-mask              255.255.255.0;
# make sure Windows clients work fine
# Hybrid node - WINS first. Then broadcast
# for WINS servers see on each sub network
# lease time = 3 hours
default-lease-time              10800;
max-lease-time                  10800;
deny unknown-clients;
use-host-decl-names             on;
# from 3.0b2pl11 - No DDNS registration
ddns-update-style               none;
# ---------------------------- GROUP DEFINITIONS -------------------------------
#
group {
  host node1 {
        # if the DHCP server is on a different machine than the Rembo Server,
        # declare in the following two lines the Rembo Server's name
        # server-identifier                       myremboserv.mydomain.org;
        # server-name                              "myremboserv";
        hardware ethernet                       00:02:b3:1a:5f:16;
        fixed-address                           192.168.1.1;
        option host-name                        "pctest";
  }
}
# ------------------------ SUBNET DEFINITIONS  -------------------------------
#
# Private networks that the DHCP should listen
#
subnet 192.168.1.0 netmask 255.255.255.0 {
   option broadcast-address   192.168.1.255;
}
#
# Public networks that the DHCP should listen
#
subnet 123.123.123.0 netmask 255.255.255.0 {
   option routers            123.123.123.99;
   option broadcast-address   12.123.123.255;
}
# end of dhcpd.conf
```

8.4 dhcpd.c source code for the /etc/init.d/dhcpd used at the ESRF for RedHat 7.1 servers

```
/* This is a wrapper program to execute dhcpd.sh script */
#include <sys/wait.h>
int main(int narg,
         char *argv[])

{
  char command[128];
  char buff[80];
  int i,rc;
  strcpy(command,"/etc/init.d/dhcpd.sh ");
  strcat(command,argv[1]);
  /* get stdin and stdout, stderr OK */
  setuid(0);
  rc=system(command);
  if ( WIFEXITED(rc) != 0 ) {
    return(WEXITSTATUS(rc));
  }
  else {
    return(0);
  }
}
```

8.5 Initial Rembo configuration file used at the ESRF

```
# ESRF Rembo NBP config file
# Modified: $Date: 2002/03/05 15:04:23 $

# BaseDir <string>
# Specifies the home dir for the server. All paths can then be
# specified as relative to this base directory
# e.g. Basedir "c:/bootrom/rembo"
BaseDir "/opt/rembo"

# NetPassword <string>
# This password will protect your server against illegal access
# to the server's files through netclnt
# This option is mandatory
NetPassword "rembo"

# Interfaces <ip-addresses>
# Specify the server IP addresses on which you want Rembo to
# receive and send packets
# When not specified, Rembo uses the IP address bound to the host name
#Interfaces 192.168.1.1

# ESRF DCHP servers will provide the initial PXE answer
DisableDHCPProxy
BootNoMulticastDiscovery
# ESRF: We allow the UDP and MCAST datagrams to cross just one router
#       Note that other, client side limitations may have been programmed
#       to the initial StartPage file.
FileMCASTTTL 2
MTFTPMCastTTL 2

AuthLocalDomain remboauth {
  UserGroup "comp"
}

# Collect all non-defined hosts here
GROUP Default {
    Options unicast
```

```
    AuthDomain "remboauth"
    StartPage "net://global/rembowiz_nodefs.shtml"
}
#
# end of rembo.conf
```

8.6 /etc/init.d/rembo.sh used at the ESRF for RedHat 7.1 servers

```
#!/bin/sh
#
# Sample init script for starting REMBO automatically on boot-up
#
# For RedHat Linux 7.x
#

# This is the path where REMBO is installed
#
REMBODIR=/usr/local/rembo
cmdline="${REMBODIR}/rembo -v 3 -c ${REMBODIR}/rembo.conf"

# Source function library.
. /etc/rc.d/init.d/functions

check_that_running() {      # verify that the named process(es) running, else error
        pid=`/bin/ps -ef |
              /bin/grep $1 | /bin/grep -v $0 | /bin/grep -v $$ |
                /bin/grep -v "rembo start" |
                /bin/grep -v "rembo stop" |
                /bin/grep -v "rembo restart" |
                /bin/grep -v "rembo reload" |
                /bin/grep -v "rembo.sh start" |
                /bin/grep -v "rembo.sh stop" |
                /bin/grep -v "rembo.sh restart" |
                /bin/grep -v "rembo.sh reload" |
                /bin/grep -v "tina_" |
                /bin/grep -v "emacs " |
                /bin/grep -v "vi " |
                /bin/grep -v "view " |
              /bin/grep -v grep | /bin/awk '{print \$2 }'   `
       echo "    check_that_running():"
        if [ "$pid" = "" ]
       then
              echo "    - no process(es) named $1 running"
              echo -n "    - error exit from script"
              echo_failure
              echo ""
              exit 1
       else
              echo "    - named process(es) $1 are running ($pid)"
       fi
}

check_that_not_running() {      # verify that the named process(es) not there
        pid=`/bin/ps -ef |
                /bin/grep $1 | /bin/grep -v $0 | /bin/grep -v $$ |
                /bin/grep -v "rembo start" |
                /bin/grep -v "rembo stop" |
                /bin/grep -v "rembo restart" |
                /bin/grep -v "rembo reload" |
                /bin/grep -v "rembo.sh start" |
                /bin/grep -v "rembo.sh stop" |
                /bin/grep -v "rembo.sh restart" |
                /bin/grep -v "rembo.sh reload" |
                /bin/grep -v "tina_" |
```

```
                /bin/grep -v "emacs " |
                /bin/grep -v "vi " |
                /bin/grep -v "view " |
                /bin/grep -v grep | /bin/awk '{print \$2 }'  `
        echo "    check_that_not_running():"
        if [ "$pid" != "" ]
        then
                echo "    - named process(es) $1 still running ($pid)"
                echo -n "    - error exit from script"
              echo_failure
              echo ""
                exit 1
        else
                echo "    - no process(es) named $1 running"
        fi
}

killmasterprocess() {       # finds out the process forked by daemon(), kills it
       pid=`/bin/ps -ef | /bin/grep $1 | /bin/grep -v $0 | /bin/grep -v $$ |
                /bin/grep -v "rembo start" |
                /bin/grep -v "rembo stop" |
                /bin/grep -v "rembo restart" |
                /bin/grep -v "rembo reload" |
                /bin/grep -v "rembo.sh start" |
                /bin/grep -v "rembo.sh stop" |
                /bin/grep -v "rembo.sh restart" |
                /bin/grep -v "rembo.sh reload" |
                /bin/grep -v "tina_" |
                /bin/grep -v "emacs " |
                /bin/grep -v "vi " |
                /bin/grep -v "view " |
              /bin/grep -v grep |  awk '{printf ("parent%s %s\n",\$3,\$2);}' |
              grep "parent1 " | awk '{print \$2}' `
       echo "    killmasterprocess():"
        if [ "$pid" = "" ]
       then
              echo "    - could not find process(es) named $1 forked by (1)"
       else
              echo "    - kill(1):signal($2),process($pid),named($1),forked by (1)"
              kill $2 $pid
       fi
}

echo "/etc/init.d/rembo:"
case "$1" in
'start')
       check_that_not_running rembo
       echo "    - Starting rembo with commandline"
       echo -n "      $cmdline"
       daemon $cmdline > /dev/null 2>&1
       sleep 2
       echo ""
       check_that_running rembo
       ;;
'stop')
       killmasterprocess rembo -TERM
       sleep 2
       check_that_not_running rembo
       ;;
'reload')
       check_that_running rembo
       killmasterprocess rembo -HUP
       sleep 2
       check_that_running rembo
```

```
        ;;
'restart')
        killmasterprocess rembo -TERM
        sleep 2
        check_that_not_running rembo
        echo "    - Starting rembo with commandline"
        echo -n "      $cmdline"
        daemon $cmdline > /dev/null 2>&1
        sleep 2
        echo ""
        check_that_running rembo
        ;;
*)
        echo -n "    usage: rembo {start|stop|restart|reload}"
        echo_failure
        echo ""
        exit 1
        ;;
esac
echo -n "    exiting - no errors"
echo_success
echo ""
exit 0
```

8.7 *rembo.c source code for the /etc/init.d/rembo used at the ESRF for RedHat 7.1 servers*

```
/* This is a wrapper program to execute rembo.sh script */
#include <sys/wait.h>
int main(int narg,
         char *argv[])

{
  char command[128];
  char buff[80];
  int i,rc;
  strcpy(command,"/etc/init.d/rembo.sh ");
  strcat(command,argv[1]);
  /* get stdin and stdout, stderr OK */
  setuid(0);
  rc=system(command);
  if ( WIFEXITED(rc) != 0 ) {
    return(WEXITSTATUS(rc));
  }
  else {
    return(0);
  }
}
```

✍