

# Replicator 3.0 for Debian GNU/Linux 3.0 (woody)

## User's Guide

Sébastien Chaumat

15th December 2002

## Contents

<b>1</b>	<b>Conventions</b>	<b>3</b>
<b>2</b>	<b>Disclaimer</b>	<b>3</b>
2.1	Authors . . . . .	3
2.2	Your work . . . . .	3
2.3	License . . . . .	3
2.4	Greetings . . . . .	3
2.5	Support . . . . .	4
<b>3</b>	<b>Introduction</b>	<b>5</b>
3.1	Where to find it . . . . .	5
3.2	Purpose . . . . .	5
3.3	Overview of the replication . . . . .	5
3.4	Warnings . . . . .	5
3.5	What's new since 2.X . . . . .	6
3.6	TODO . . . . .	6
<b>4</b>	<b>Using Replicator</b>	<b>7</b>
4.1	Installing Replicator (warning) . . . . .	7
4.2	Upgrading from 1.X or 2.X . . . . .	7
4.3	Configuration of <i>replicator.conf</i> . . . . .	7
4.3.1	the general section . . . . .	7
4.3.2	the miniroot section . . . . .	7
4.3.3	the bootdisk section . . . . .	7
4.3.4	the target network section . . . . .	8
4.3.5	the target partitioning section . . . . .	8
4.3.6	the update rules section . . . . .	8
4.3.7	the miscellaneous section . . . . .	9
4.3.8	the Experts Only section . . . . .	9
4.4	Site preparation for the replication . . . . .	9
4.4.1	Compiling a boot kernel for the installation floppy . . . . .	9
4.4.2	Preparing the miniroot . . . . .	10
4.4.3	Authorizing targets to access model with rsyncd . . . . .	10
4.4.4	Preparing the custom post-replication script . . . . .	11
4.5	Installing the target . . . . .	11
4.5.1	Creation of the boot floppy . . . . .	11
4.5.2	Using the boot floppy . . . . .	11

4.6	The light classes mechanism (aka the strategy guide) . . . . .	12
4.6.1	Installing computers in more than one network . . . . .	12
4.6.2	Installing a special computer among a set of identical others . . . . .	12
<b>5</b>	<b>Doing more with Replicator</b>	<b>14</b>
5.1	Modifying the behavior of replicator with the command line . . . . .	14
5.2	Keeping 2 computers synchronized . . . . .	14
5.3	Installing a diskless client . . . . .	14
5.4	Installing a client with <i>/usr</i> mounted by NFS . . . . .	14
5.5	model being different of miniroot-server . . . . .	15
<b>A</b>	<b>Default update rules</b>	<b>16</b>
<b>B</b>	<b>replicator.conf.example</b>	<b>18</b>

# 1 Conventions

- This is a **computer**
- This is a `command`
- This is a `$perl_scalar_variable`
- This is a `@perl_list_variable`
- This is a *file*

# 2 Disclaimer

## 2.1 Authors

Replicator is an original work by Loïc Prylli (lprylli@lhpc.univ-lyon1.fr). It originated as some custom scripts to help quickly installing computer classrooms and or clusters of computers. As I (Sébastien Chaumat (schaumat@ens-lyon.fr)) needed to do the same kind of work (having the same Debian/GNU Linux system on a lot of different computers), we started to transform the initial scripts to make a general-purpose system allowing to quickly and non-interactively install computers in any networked environment. It allows you to choose for each station whether it will be a full install, a “/usr nfs-mounted” install or a diskless install. It can handle automatically the main steps of hardware detection (hard disk, Xfree86), and partitioning (either a completely automatic partitioning with the option of preserving your DOS/Windows partition, or a manual partitioning preserving some of the hard disk filesystems).

## 2.2 Your work

If you use Replicator please post your comments to the public open discussion forum at SourceForge:

[http://sourceforge.net/forum/forum.php?forum\\\_id=36345](http://sourceforge.net/forum/forum.php?forum\_id=36345)

## 2.3 License

Copyright (C) 1999-2002 by Sébastien Chaumat (schaumat@debian.org) and Loïc Prylli (lprylli@lhpc.univ-lyon1.fr)

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

A copy of the GNU General Public License is available as `/usr/share/common-licences/GPL` in the Debian GNU/Linux distribution or on the World Wide Web at <http://www.gnu.org/copyleft/gpl.html>. You can also obtain it by writing to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

## 2.4 Greetings

Pascal Degiovani (degio@ens-lyon.fr) convinced me to look for an auto-install tool for Linux. He introduced Loïc Prylli to me. I’m very grateful to him.

I thanks the École Normale Supérieure de Lyon for its support.

Bertrand Louis-Lucas proof-read this documentation.

Dominique Ponsard and Hervé Brunet made beta-testing.  
Lots of debugging was done by Bernd Harmsen and Jerome Warnier.  
Thanks to all users who reported bugs and made suggestion.

## **2.5 Support**

Limited support is available in the public area at:

`http://sourceforge.net/projects/replicator/`

We would be really pleased to help you.

## 3 Introduction

### 3.1 Where to find it

You can find the latest version of Replicator at :

<http://replicator.sourceforge.net>

Check out the cvs for the very last improvements.

### 3.2 Purpose

If you are the system administrator of a networked site, how can you (re)install quickly a computer in generally less than five minutes (even less if you install a set of similar machines)?

You want to customize the configuration of a computer only once (on the **model** computer) for your site. Moreover you want any new **target** computer to have almost the same configuration as the **model**. The **target** will be mostly identical to the **model** except:

- some specific configuration files (network, fstab, X configuration, hostname),
- the way it is partitioned,
- the possibility of being diskless, having /usr NFS-mounted instead of local, or having specific filesystems (local homes for instance),
- ...

The present software, called “Replicator”, will automate the installation of **target**. More precisely it consists of a set of easy-to-use scripts to prepare the installation, the result of the preparation being a floppy that when booted on the target, will almost automatically install it (by default, it asks confirmation first!).

### 3.3 Overview of the replication

The copying process is composed of the following steps:

1. Installation and configuration of Replicator on the computer **miniroot-server** which usually is also the **model** you want to replicate,,
2. Creation and setup of a tiny NFS root filesystem onto **miniroot-server**,
3. Creation of bootfloppy onto **miniroot-server**,
4. Configuration of remote access to the **model**,
5. Quick install of the computer **target** with the floppy.

### 3.4 Warnings

1. The whole directory tree of your **model** machine will be duplicated according to the “update rules” mentioned in 4.3.6. If you created non-standard directories (e.g. mountpoints) which you don’t want to duplicate, then edit the update rules accordingly.
2. This software is experimental (but tested and used in production environment), be sure that for every machine where you execute one of the scripts or for every machine that you include in one of the configurations files, all the volumes accessible from the machine (either local or NFS-mounted) are properly backup’ed,
3. At this point, it is recommended that you understand a bit of perl and sh scripts, to be able to diagnose potential problems.

### 3.5 What's new since 2.X

- All networks are defined in *replicator.conf*: no more need for a bootdisk per network (see 4.3.3).
- The miniroot is made with debootstrap.
- You can define the filesystem type you want on the target (actually the miniroot contains packages needed for reiserfs,jfs,xfs. Nothing else than kernel support is needed for ext3).
- You can set the keymap list to choose from at replication time.
- lilo.conf can be generated on the target using a template.
- The localisation of replicator's script is the same on the miniroot server and inside the miniroot.
- At the beginning of the replication on the target, the media detected by replicator for installation is displayed with the partitioning/mksf choices.
- WARNING : To actually modify the update\_rules you have to set the \$modify\_rules\_file (see 4.3.6).

### 3.6 TODO

What we are working on:

- read-only miniroot with ramdisk support for parallel replication
- add rsyncd password / ssh encryption for replication of sensitive data
- ultra fast replication using an archive on model
- faster than light replication for clusters using codafs
- switch to full class mechanism.
- make congelator: put the model on a CD.
- PXE helper

## 4 Using Replicator

The computer you want to replicate is called your **model**. All along this documentation we refer to a computer called **miniroot-server**. For basic use of replicator, **miniroot-server** *IS THE SAME* than **model**. This computer is the one where you want to install replicator.

### 4.1 Installing Replicator (warning)

It's a standard Debian package :-). Just use dpkg or apt.

*Warning:* The kernel running on the computer where you will create the bootdisk (see 4.5.1) must support loop device. Standard kernels shipped with Debian are ok.

### 4.2 Upgrading from 1.X or 2.X

If you are upgrading from version 2.X, I strongly suggest that you create a new miniroot using `repli-miniroot`. You should also check carefully the new `/usr/share/doc/replicator/replicator.conf.example`.

### 4.3 Configuration of *replicator.conf*

Edit the file `/etc/replicator/replicator.conf` (it is heavily commented). Change the value of the perl variables to fit your needs.

The following sections are presented here in the order they are used during the replication.

#### 4.3.1 the general section

- `$verbose`: set this to 1 and replicator will be more verbose than usual.
- `$debian_version`: set this to 2.2 to replicate an old potato model.

#### 4.3.2 the miniroot section

- `$nfsroot`: where, on miniroot-server, will you create the miniroot.

#### 4.3.3 the bootdisk section

- `$bootkernel`: full path to the *bzImage* you prepared as explained in 4.4.1.
- if you need to boot one target on a serial console:
  - `$grub`: set to 0 to use lilo instead of grub on the bootdisk. This is compulsory if you want to boot on a serial console (grub serial boot is not supported in replicator yet).
  - `$serial`: definition of serial port for lilo, only if you want to boot on serial console,
  - `$serialcons`: definition of serial port for the kernel, only if you want to boot on serial console.
- `@networks`: define one network per line. For each network, choose a name for the target list then create the corresponding perl variable. Look at the following example:

```
#List of networks/targets the bootdisk will handle.
#you don't need to fill this variable if you only
#plan to install dhcp clients.
```

```
@networks =
```

```
#   network          domain          netmask          gateway          broadcast  targets
```

```
(["192.168.32.0","prof.my.domain","255.255.255.0","192.168.32.1","192.168.32.255","teachers"]
["192.168.33.0","my.domain","255.255.255.0","192.168.33.1","192.168.33.255","sysadm_net"]
["192.168.34.0","maths.my.domain","255.255.255.0","192.168.34.1","192.168.34.255","maths"]
);

#for each network create the corresponding targets list
@teachers=qw(teach1 teach2);
@sysadm_net=qw(adm1 adm2 adm3);
@maths=qw(maths1 maths2 maths3 maths4);
```

#### 4.3.4 the target network section

- The following variables are only used for non dhcp targets:
  - \$gateway: name or IP address of the gateway of **target** (can be empty).
  - \$netmask: e.g. '255.255.255.0'.
  - \$network: e.g. '192.168.32.0'.
  - \$broadcast: e.g. '192.168.32.255'.
  - \$domainname: e.g. 'my.fake.domain'.
- \$hosts\_supp: lines to add to the */etc/hosts* of target machine.
- \$nfsopts: for diskless stations or NFS-mounted */usr* partition only. See 5.3.

#### 4.3.5 the target partitioning section

- \$target\_disk: disk of **target** on which you want to install. By default replicator use the first detected disk (hda or sda).
- \$mkfs AND \$fs : set *both* of this variables to use another filesystem type on the **target**. The kernel you use on the replication floppy must support this filesystem. The miniroot contains needed packages for reiserfs, xfs and jfs. e.g.
 

```
$mkfs="mkfs -text3"
$fs="ext3"
```
- \$custom\_part: set to 1 if you want to manually partition the hard disk of **target**,
- @autopart\_specs: specification of the partitioning scheme of \$target\_disk (min and max size for each partition). See *replicator.conf.example*.
- @manualpart\_specs: describe the existing partitions and filesystems to reuse. Only used if \$custom\_part=1. See *replicator.conf.example*.
- \$fstab\_supp: lines to add to the */etc/fstab* of **target**.

#### 4.3.6 the update rules section

Nothing but a reminder. To fine tune the list of files/directories which are replicated, edit */etc/replicator/update\_rules* AND set \$modify\_rules\_file to 1. you delete this file, the default rules are in */usr/lib/replicator/update\_rules.default*. In this files are defined 3 variables:

- @slash\_exclude



- @usr\_exclude
- @var\_include

This is how replicator proceeds:

- It replicates the root (/) filesystem except for what is mentioned in @slash\_exclude. If you use cfengine (a very useful program for network administration), replicator parses */etc/cfengine/cfengine.conf* and does not copy files handled by cfengine copying facility.
- It replicates the */usr* directory (not the filesystem) except for what is mentioned in @usr\_exclude. Be aware that it may cross filesystems to achieve this.
- It replicates the directories structure of */var* (only directories, no files).
- It replicates the contents of the */var* subdirectories mentioned in @var\_include.

To define an empty update rule use the syntax: @usr\_exclude=qw("");

#### 4.3.7 the miscellaneous section

- \$usr\_is\_nfs: set this variable to 1 if */usr* must be a NFS-mounted filesystem on **model** instead of being on a local partition (see 5.3).
- @extra\_keymaps: this is the list of keymaps you will be asked to choose from when booting the target at replication time. Note that the keymap of the **model** is automatically added.

#### 4.3.8 the Experts Only section

- \$model : if the **model** is NOT the same as the **miniroot-server** set this to the name of the **model** and read 5.5.
- \$lilo\_template : instead of generating the */etc/lilo.conf* file, replicator can use the one you provide here. Only the boot and root statements will be changed by replicator according to the detected medium on the target.
- \$skip\_lilo : if you don't want replicator to generate *lilo.conf*, the set this to 1. To force replicator to copy the */etc/lilo.conf* from the model to the target, you must modify the update rules (*lilo.conf* is excluded by default). See 4.3.6.
- \$skip\_fstab : same as \$skip\_lilo but for */etc/fstab*.
- \$sharedir, \$scriptsdir, \$sbindir : used to run replicator from the cvs . See */usr/share/doc/replicator.conf.examples*

### 4.4 Site preparation for the replication

#### 4.4.1 Compiling a boot kernel for the installation floppy

Compile a kernel with all the necessary driver for the computer **target**. Do not use modules for the drivers required at boot time (typically network and SCSI). You should even completely disable modules support in this kernel.

Add the following option during configuration of the kernel:

- Networking options / IP: kernel level autoconfiguration
- Networking options / IP: kernel level autoconfiguration / DHCP support

- Filesystems / Network File Systems / NFS filesystem support
- Filesystems / Network File Systems / NFS filesystem support / Root file system on NFS
- Filesystems/ \* : add the filesystem you want to put onto the partitions of the targets (see the \$mkfs var in 4.3.5)

You *must* create such a kernel because the standard Debian kernels do not have nfsroot nor dhcp support enabled. This kernel will only be used for the installation floppy and will not be installed on **target**.

Compile this kernel with classical `make dep ; make bzImage` (don't use `make-kpkg`). This kernel will be use to boot the computer **target**. Put this new kernel in a safe place.

#### 4.4.2 Preparing the miniroot

As root, execute `repli-miniroot`. You need acces to a Debian potato mirror. It can be a CD set, a remote mirror somewhere on the network or a local mirror.

This will create a special filesystem (e.g. `/export/install/miniroot`) whith a basical Debian/GNU Linux on it.

**VERY IMPORTANT:** Each time you modify the configuration in `/etc/replicator/` you must run the command:

```
repli-miniroot --update-config
```

or

```
repli-miniroot -u
```

Don't forget to export `miniroot-dir` for **target** with the `no_root_squash` option. To do that, edit `/etc/export` and add a line like:

```
/export/install/miniroot target.my.domain(rw,no_root_squash)
```

Then tell the nfs-server to reload it's configuration:

```
/etc/init.d/nfs-server reload
```

**Warning:** Every line of `/etc/export` file must end with a newline character otherwise you will get silly error messages at boot time.

#### 4.4.3 Authorizing targets to access model with rsyncd

Replicator comes with a `/etc/replicator/rsyncd.conf` file:

```
#
#this is rsyncd.conf for replicator
#
[replicator]
path=/
use chroot=no
read only=yes
uid=0
#max connections = 20
#
#in case you don't use tcpwrapper (bad:)
hosts allow=192.168.0.0/255.255.0.0
hosts deny=*
```

You are responsible for launching rsyncd on the **model** with this config files. That means either moving this file to `/etc/rsyncd.conf` or editing your existing `/etc/rsyncd.conf` to add the rsync module [replicator] as defined into `/etc/replicator/rsyncd.conf`.

You can either :

- configure inetd to allow rsyncd.
- make a script to launch rsyncd as a standalone server at boot time.
- launch rsyncd by hand with the command `rsync -daemon`.

Don't forget to change the `hosts allow` and `hosts deny` settings.

#### 4.4.4 Preparing the custom post-replication script

You can create a shell script `/etc/replicator/repli-postint`. It will be executed at the very end of the copying process. It is *your* postint script. Again run `repli-miniroot -update-config` to copy this file to the miniroot.

*Note:* This script is run chrooted in the root filesystem of **target** (exactly as it would run after rebooting). Here are some suggestions of what can you put in this script:

- `dpkg-reconfigure ssh`: will create a new public/private keys pair.

### 4.5 Installing the target

#### 4.5.1 Creation of the boot floppy

Use the script `repli-bootdisk`.

You can put a floppy right now in your floppy drive. If you interrupt `repli-bootdisk` with `ctrl-C` you can copy by hand the file `/tmp/name-of-the-target.img` on a floppy with the command `cp`.

#### 4.5.2 Using the boot floppy

1. sit down on front of the **target** computer,
2. insert the floppy,
3. turn the power on,
4. login as root with the password you set at the creation of the miniroot,
5. answer the 4 questions,
  - (a) `repli-install hdsetup (-real)`
  - (b) `repli-install hostconf (-real)`
  - (c) `repli-install netcopy (-real)`
  - (d) `repli-install configure(-real)`
  - (e) `repli-postint`
7. remove the floppy,
8. reboot.

## 4.6 The light classes mechanism (aka the strategy guide)

This is very important and usefull. One easy way to define replication classes is to create specific configuration files.

To summarize:

- `repli-bootdisk -config /etc/replicator/my_specific_file` first loads `/etc/replicator/replicator.conf` and then loads the file `my_specific_file`. Whatever you put into `my_specific_file` overrides the content of `/etc/replicator/replicator.conf`.
- *after booting the target* the file `/etc/replicator/replicator.conf` is read first and, if it exist, the file `/etc/replicator/replicator.conf_targetname` is read.
- `/etc/replicator/replicator.conf_targetname` can be a symbolic link to `/etc/replicator/replicator.conf_myclass`.

The following examples should help you understand how replicator deals with its configurations files.

### 4.6.1 Installing computers in more than one network

Starting with version 2.0.1 of replicator, you only need to fill the variable `@networks` and the corresponding lists of targets in `replicator.conf`. See 4.3.3.

### 4.6.2 Installing a special computer among a set of identical others

Lets suppose you are installing a classroom: all computers are identical (call them **student1**, **student2**, ...) except the teacher's one (just call it **teacher**).

For example, students' computers, have a 1GB harddisk with `/home` nfs-mounted and boot with dhcp, whereas **teacher** has hardcoded network configuration, one 4GB disk for the OS and a second ide disk with local `/home` in `/dev/hdb1`.

In such a case we use `/etc/replicator/replicator.conf` to put defaults for the identical computers and we make a file `/etc/replicator/replicator.conf_teacher` for the unique computer **teacher**.

In `/etc/replicator/replicator.conf` :

```
@autopart_specs =
#mount point          min_size      max_size
([ "/" ,              "100Mo",      "1000Mo"],
 [ "swap",            "64Mo",       "128Mo"],
);
$fstab_supp = "/dev/fd0 /floppy auto defaults,user,noauto 0 0 homeserver:/exports/home /home"
```

In file `/etc/replicator/replicator.conf_teacher`:

```
\begin{verbatim}
@targets=qw(teacher);
$domainname = 'classroom.here';
$gateway = '192.168.32.23';
$netmask='255.255.255.0';
$network='192.168.32.0';
$broadcast='192.168.32.255';
#mount point          min_size      max_size
([ "/" ,              "100Mo",      "400Mo"],
 [ "swap",            "64Mo",       "128Mo"],
\end{verbatim}
```

```

[ "/usr" ,          "800Mo" ,      "3000Mo" ],
[ "/var" ,          "200Mo" ,      "1000Mo" ],
);
$fstab_supp = "/dev/fd0 /floppy auto defaults,user,noauto 0 0
/dev/hdb1 /home auto defaults 0 0
/dev/cdrom /cdrom auto ro,defaults,user,noauto 0 0\n";

```

Now create the bootisk with :

```
repli-bootdisk -config /etc/replicator/replicator.conf_teacher
```

To install a student computer, boot it with the bootdisk and choose “Boot with DHCP and Install a DHCP target” in grub’s menu. To install **teacher**, use the same bootdisk and choose “Boot Teacher ...” in grub’s menu.

## 5 Doing more with Replicator

### 5.1 Modifying the behavior of replicator with the command line

You can also give the arguments on the command line of the different scripts `repli-command`: instead of "`$var = val`" in *replicator.conf* you can use the option `--var val`. Example:

```
repli-bootdisk -bootkernel /path/to/the/boot/kernel
```

Command-line arguments take precedence over the configuration file *replicator.conf*.

### 5.2 Keeping 2 computers synchronized

Suppose you change something (e.g. you add a package) onto the computer **model**. To keep **target** a real copy of **model** you can either make the same operation by hand onto the computer **target** or use the script `repli-sync`.

To use this script you must install replicator onto the target (it is already on the target if it was installed onto the model).

Before using `repli-sync [-real]`, you have to check that the file */etc/replicator/reply-sync.conf* exists on the target.

It should contains something similar to:

```
$model="mymodel";
@usr_exclude=qw(/var /local /src);
$noboot=1; #option : do not update /boot
$norcd=1; #option : do not update /etc/rc?.d
$nouser=0; #option : do not update /usr
#do not remove the following line..
$the_end=1;
```

### 5.3 Installing a diskless client

If you want to easily install a diskless client, **target**, with its filesystem on the computer **miniroot-server**, follow these instruction:

- export with NFS, read-only, the directory */usr* of **miniroot-server** for itself and for **target**,
- edit the variable `$targetbase` into `repli-diskless` or create a directory */export/diskless*,
- create the NFS root filesystem of **target** by en executing the command: `sh repli-diskless`,
- export with NFS */export/diskless/target* for **target**,
- edit *repli-postint.diskless* if necessary,
- create the bootdisk of **target** by executing the command:  
`repli-bootdisk -nfsroot /export/diskless/target -host target`

Boot **target** with this floppy. That's all!

### 5.4 Installing a client with */usr* mounted by NFS

Add the option `$usr_is_nfs=1` into *replicator.conf* and export the directory */usr* of **model** for **target**.

## 5.5 model being different of miniroot-server

You may want to have a dedicated **miniroot-server**, for security reason or because you just don't want to install replicator onto your **model**.

To do so, first install replicator on **miniroot-server**. Then in */etc/replicator/replicator.conf* customize the variable `$model`.

Then read again this documentation without considering **miniroot-server=****model**.

## A Default update rules

You can find them in `/usr/lib/replicator/update_rules.default`.

```
#
#update_rules
#
#In this file you can tune the behaviour of replicator
#
#WARNING: to define an empty rule use the syntax: @usr_exclude=qw("");
#####
#                               #
# "/" exclude rules #
#                               #
#####
#The following files/directoriess are NOT copied from $server to $target

@slash_exclude=qw(
/etc/my
/etc/network/interfaces
/etc/.rootkey
/etc/adjtime
/etc/isapnp.conf
/etc/minirc*
/etc/gpm.conf
/etc/fstab
/etc/lilo.conf
/etc/hostname
/etc/hosts
/etc/aboot.conf
/etc/exports
/etc/dhcpd.conf
/etc/fmirror
/etc/apache/*
/etc/squid.conf
/etc/chatscripts/*
/etc/init.d/network
/etc/ppp/*
/etc/named.conf
/etc/named.boot
/etc/ssh/ssh_host_*key*
/etc/ssh/ssh_random_seed
/etc/rmtab
/etc/mtab
/etc/kbd/*
/etc/console-tools/*
/etc/X11/XF86Config*
/etc/X11/Xserver
/etc/ioctl.save
/etc/mon/
/etc/ssh2/hostkey
```



```

*.cfsaved
*.dpkg-old
/tmp/*
/var
/usr
/bigpart
/boot
/*old*
/cdrom/*
/local
/localhd
/fatboot
/home*
/proc/*
/tchich
/ramdisk/
/linuxppc/
/oldexports/
/amd/
/tftpboot/
/*mnt*/
/floppy/*
/export*/
/exports/*
/opt*
/net*
/vol*
/ufs*
/lost+found/
);

```

```

#####
#                               #
#/usr exclude rules #
#                               #
#####
#The following files/subdirectories of /usr are NOT copied from $server to $target

```

```
@usr_excl= qw(local src);
```

```

#####
#                               #
#/var include rules #
#                               #
#####
#ONLY the content of the following subdirectories of /var will be copied from $server to $target
#NOTE: the complete directories structure will be replicated (not the content of those directories)
@var_include = qw(lib);
#NOTE: it's not possible to always include yp/nicknames here. If this file
#does not exist, rsync will fail.

```

## B replicator.conf.example

You can find this file in `/usr/share/doc/replicator/`.

```
#/etc/replicator/replicator.conf
#
#This is the place to configure replicator

#Warning this file is an example. Customize it and check it twice
#before using replicator.

#####
#
# Section: general configuration
#
#####

#Be very verbose
$verbose=1;

# Debian version to install both inthe miniroot and on the target
#(2.2 for potato, 3.0 for woody/testing)
#comment out to let replicator guess for you

$debian_version=3.0;

#####
#
# Section: miniroot configuration
#
#####

#Where will you create the miniroot
$nfsroot= "/export/miniroot";

#####
#
# Section: bootdisk configuration
#
#####

# If for the bootdisk you want to use a special kernel rather
# than the Debian default.
# Debian default is not yet supported
$bootkernel = "/etc/replicator/bzImage-for-replicator";

#not used yet
#$initrd = '';

#to boot on a console on a serial port customize this:
#$grub=0; #we need to use lilo instead
```

```

#$serial='0,9600n8';
#$serialcons='console=ttyS0,9600 CONSOLE=/dev/ttyS0';

#List of networks/targets the bootdisk will handle.
#you don't need to fill this variable if you only
#plan to install dhcp clients.

@networks =
#   network          domainname          netmask          gateway          broadcast          n
(
#   [ "192.168.32.0" , "prof.my.domain",  "255.255.255.0", "192.168.32.1", "192.168.32.255", "te
#   [ "192.168.33.0" , "my.domain",      "255.255.255.0", "192.168.33.1", "192.168.33.255", "sy
#   [ "192.168.34.0" , "maths.my.domain", "255.255.255.0", "192.168.34.1", "192.168.34.255", "ma
);

#for each network create the corresponding targets list
#@teachers=qw(teach1 teach2);
#@sysadm_net=qw(adm1 adm2 adm3);
#@maths=qw(maths1 maths2 maths3 maths4);

#####
#                                     #
# Section: network configuration of the target #
#                                     #
#####

#Lines to add to the /etc/hosts of the target computer
#$hosts_supp = "192.168.32.1  host1.myfake.domain host1 \n
192.168.32.45  mynfserver.myfake.domain mynfserver \n";

# NFS options for diskless stations or NFS-mounted /usr partition
#$nfsopts = "nolock,rsize=8192,wsiz=8192";

#####
#                                     #
# Section: partitioning of the target #
#                                     #
#####

# To install on a specific disk rather than the first detected disk,
# modify and uncomment next line
#$target_disk = "hdb" ;

# For MANUAL partionning set $custom_part to 1
$custom_part = 0;

# Command to create filesystem (not for swap)
# The filesystem type must be compiled in the kernel you
# put on the bootdisk ($bootkernel)

```

```

# you must fill the 2 vars in a consistent way
$mkfs="mkfs -text3";
$fs="ext3";

#For AUTOMATIC PARTITIONING (not used if $custom_part = 1)
@autopart_specs =
#mount_point      min_size max_size
([ "/" , "100Mo","100Mo"],
 [ "swap", "64Mo","128Mo"],
 [ "/usr" , "800Mo",      "3000Mo" ],
 [ "/var" , "200Mo",      "500Mo" ],
 [ "/export/home", "remaining" ]
);

#For MANUAL partitioning (not used if $custom_part = 0)
# for instance if you want to keep an existing partition
# or to define the partition sizes manually before starting the install
# process, please gives the affectation of filesystems to partition numbers
# and which ones need to be allocated

@manualpart_specs =
#mount_point partition_number mkfs needed?(0|1)
([ "/" , 1, 1],
 [ "swap" , 2, 1],
 [ "/usr" , 3, 1],
 [ "/var" , 5, 1],
 [ "/export/home", 6, 0]
);

#Lines to add to the /etc/fstab of the target machine
#This will not be used if $skip_fstab is used

$fstab_supp = "/dev/fd0 /floppy auto defaults,user,noauto 0 0
#myfnsserver:/usr/local /usr/local nfs ro,defaults,rsize=8192,wsiz=8192 0 0
/dev/cdrom /cdrom auto ro,defaults,user,noauto 0 0\n";

#####
#
# Section: update rules
#
#####

#To fine tune the list of files/directories which are
#replicated, edit /etc/replicator/update_rules (default rules
#are in /usr/lib/replicator/update_rules.default) and uncomment
#the following line

#$modify_rules_file = 1;

```

```
#####
#
# Section: miscellaneous
#
#####

# To have an NFS /usr-mounted partition on the target
# $usr_is_nfs=1;

# List of keymaps to choose from on the copy (see in /usr/share/keymaps/*/*/)
# The default choice is to use the keymap of the model
# Use the syntax : "keymap-name","description"
# @extra_keymaps=("fr-latin0","french keyboard","us","us keyboard","uk","uk keyboard");

#####
#
# Section: Experts Only
#
#####
# Name of the model machine. Change this only if
# the model machine is not the miniroot server
# $model= "mymodel";

# if you want a lilo template (which will just be prefixed by the
# right boot and root command, put it here)

# $lilo_template = "/somewhere/a-lilo-file";

# if you want to manage lilo.conf manually (or in repli-postinst)
# $skip_lilo = 1;

# if you want to manage /etc/fstab manually (or in repli-postinst)
# $skip_fstab = 1;

# to run from cvs, put the configuration file in /etc/replicator/
# and uncomment this
# $sharedir='./';
# $scriptsdir='./';
# $sbindir='./';

#Do NOT modify the following line AND leave it at the end of this file
$the_end=1;
```