Burning a RedHat CD HOWTO

Luigi Bitonti

<uknadors (at) yahoo (dot) com>

Morten Kjeldgaard

<>

Peter von der Ahé

<>

Guillaume Lelarge – Review and French translation.

Revision History

Revision v2.1	2003–10–17	Revised by: lb
Added RedHat 9. Fixed some minor bugs. Thanks to all the people who have sent in comments and patches.		
Revision v2.03	2003–03–10	Revised by: lb
Added some comments and fixes to the howto. The anaconda updates are now included correctly even for versions >= 7.x.		
Revision v2.02	2003–03–06	Revised by: lb
The signature checking now works for packages targeted to versions of RedHat different from the one used to run the scripts. Corrected a bug in the version comparison section of the scripts.		
Revision v2.01	2002–12–04	Revised by: lb
Second release of the new version of the howto. All the scripts were reviewed and cleaned. The updateDist.sh script now checks that all the updates were downloaded before checking the signatures.		
Revision v2.00	2002–10–28	Revised by: lb
First release of the new version (2.00) of the HOWTO		

This document describes how to make your own CDs from different releases of the Red Hat Linux distribution (up to and including release 9), equivalent to the ones commercially available from Red Hat. The structure of the distribution is described, as well as the procedure needed to include updated RPMS into the distribution. Some hints and examples on how to customize the default installation are also presented. Scripts to automate as much as possible the (re)generation of the CD images are included. Prerequisites are a good network connection, and a CD−writer (a working knowledge of shell scripting could be really useful, though).

Table of Contents

Table of Contents

1. Introduction

There may be several reasons for making your own CDs. Perhaps you're a cheapskate and want to save the cost of the Red Hat distribution. Or, perhaps you want to include in your CDs the latest distribution with all the current updates. This is highly relevant, because after each major release of the Red Hat distribution, there have been loads of updates, several of which are security related. Just take a look at the errata page. Or maybe you want to customize the default installation adding a few packages not present in the default tree and unselecting the installation of some packages which are otherwise included in the default setup.

This is what you will be taught in the next sections (hopefully). I will use the i386 architecture and releases 7.3, 8.0 and 9 of the distribution in the examples. The notes related to the previous releases (<=6.1) were contained in a previous version of this document and were compiled by the original authors. The notes related to release 6.2 are based on tests I've not completed (and I don't know if I ever will) and some documents you can find linked in the Related documentation section. The procedure given in the following sections for RedHat 7.3 and 8.0 is likely to work on all platforms supported by Red Hat (Alpha, ppc, etc.), for all the 7.x (and maybe 8.x/9 in a not too far future) releases, but I have only tested it on the i386 platform with Redhat Linux 7.3, 8.0 and 9 (I would be interested in additional information).

The operations described have legal implications, meaning you cannot redistribute the CDs as RedHat Linux if you modify them in ways not compliant to Redhat trademark policy. To make them legally redistributable, you should always comply with the guidelines stated on the RedHat website .

Always remember to set the variables in the `rhcd.conf` file and *export* the *RHCDPATH* environment variable before running the scripts you will find throughout the rest of this document and related to releases >=6.2 of RedHat Linux. An example rhcd.conf file, which should be well commented, is provided with the scripts.

1.1. Disclaimer and License

This document 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.

Neither the author nor the distributors, or any other contributor of this document are in any way responsible for physical, financial, moral or any other type of damage incurred by following the suggestions in this text.

2. Anatomy of the Red Hat FTP site

In the spirit of the Linux community, Red Hat Software has made available their Linux distributions for several platforms on their FTP site. These are all available from the top distribution directory (pub/redhat/linux/). Let's have a look at the distribution tree.

2.1. Redhat 9 directories organization

The latest distribution is, as of this writing, available only for the i386 platform. The toplevel directory appears a bit shallow, given the presence of a single architecture. (pub/redhat/linux/9/en/os/).

```
        i386/
```

Otherwise, the toplevel directory, for releases slightly older than 9, contains distributions for the different platforms. For example, the corresponding directory for release 7.1 of Redhat Linux, is structured this way:

```
        alpha/   i386/   ia64/   ppc/   s390x/
```

The root of the i386 directory in a Redhat 9 distribution looks like this:

```
        -rwxr-xr-x    2 root     root          248 Mar 14  2003 autorun
        drwxr-xr-x    7 root     root         4096 Mar 14  2003 dosutils
        -rw-r--r--    3 root     root         6192 Mar 14  2003 EULA
        -rw-r--r--    3 root     root        18385 Mar 14  2003 GPL
        drwxr-xr-x    3 root     root         2048 Mar 14  2003 images
        drwxr-xr-x    2 root     root         2048 Mar 14  2003 isolinux
        -rw-r--r--    3 root     root         6127 Mar 14  2003 README
        -rw-r--r--    2 root     root        13052 Mar 14  2003 README-Accessibility
        -rw-r--r--    2 root     root         6686 Mar 14  2003 README.de
        -rw-r--r--    2 root     root         6990 Mar 14  2003 README.es
        -rw-r--r--    2 root     root         6492 Mar 14  2003 README.fr
        -rw-r--r--    2 root     root         6805 Mar 14  2003 README.it
        -rw-r--r--    2 root     root         7995 Mar 14  2003 README.ja
        -rw-r--r--    2 root     root         7312 Mar 14  2003 README.ko
        -rw-r--r--    2 root     root         5070 Mar 14  2003 README.pt
        -rw-r--r--    2 root     root         6613 Mar 14  2003 README.pt_BR
        -rw-r--r--    2 root     root         5879 Mar 14  2003 README.zh_CN
        -rw-r--r--    2 root     root         5892 Mar 14  2003 README.zh_TW
        drwxr-xr-x    4 root     root         2048 Mar 14  2003 RedHat
        -rw-r--r--    2 root     root        25824 Mar 14  2003 RELEASE-NOTES
        -rw-r--r--    2 root     root        29902 Mar 14  2003 RELEASE-NOTES-de.html
        -rw-r--r--    2 root     root        30409 Mar 14  2003 RELEASE-NOTES-es.html
        -rw-r--r--    2 root     root        32354 Mar 14  2003 RELEASE-NOTES-fr.html
        -rw-r--r--    2 root     root        30064 Mar 14  2003 RELEASE-NOTES.html
        -rw-r--r--    2 root     root        29925 Mar 14  2003 RELEASE-NOTES-it.html
        -rw-r--r--    2 root     root        34666 Mar 14  2003 RELEASE-NOTES-ja.html
        -rw-r--r--    2 root     root        33520 Mar 14  2003 RELEASE-NOTES-ko.html
        -rw-r--r--    2 root     root        29496 Mar 14  2003 RELEASE-NOTES-pt_BR.html
        -rw-r--r--    2 root     root        22747 Mar 14  2003 RELEASE-NOTES-pt.html
        -rw-r--r--    2 root     root        25217 Mar 14  2003 RELEASE-NOTES-zh_CN.html
        -rw-r--r--    2 root     root        26645 Mar 14  2003 RELEASE-NOTES-zh_TW.html
        -rw-r--r--    3 root     root         1910 Mar 14  2003 RPM-GPG-KEY
        -r--r--r--    1 root     root         1823 Mar 14  2003 TRANS.TBL
```

The `SRPMS` directory contains the RPMS packages in source form.

The `images` directory contains boot and drivers floppy images that can be copied to a diskette if needed. In the 9 release, there is only one boot disk image available. This boot image is named `bootdisk.img`. A secondary driver disk is required beside this one if the installation is not performed directly from a CD−ROM or HD. A `boot.iso` file has now been added to boot a machine from the cdrom drive and start (network) installations more easily (i.e. without messing up with too many floppies). See section Installation and references therein for details and consult the README file in the directory for a more detailed explanation of the various files.

The `isolinux` directory contains the files needed to boot from the CD (and to rebuild bootable CDs which work the same way). This process was moved from floppy emulation to no emulation. This helps avoiding space constraints and compatibility problems.

The `dosutils` directory contains various programs for some other operating systems which are sometimes useful to support the installation process. An explanatory README file is included also in this case.

The listing is completed by a lot of files and the `RedHat` directory. The latter is the subject of the next sections while the formers have contents which will appear straightforward by simply reading their names (perhaps apart from the EULA, or End User License Agreement).

2.2. The "RedHat" directory −− the core of the distribution

The most important part of the directory tree is rooted in the `RedHat` directory:

```
drwxr-xr-x    2 root     root        53248 Jun 14 03:15 RPMS
drwxr-xr-x    2 root     root         4096 Jun 14 04:15 base
```

The `RPMS` directory contains the major part of the Red Hat distribution consisting of a set of RPM (Redhat Package Manager) files. An RPM package typically contains binary executables, along with relevant configuration files and documentation. See the section RPM packages for more information.

The `base` directory holds different files needed during the installation process, like the `comps.xml` file, which defines the *components* (groups of packages) used during the "Choose packages to install" phase. See section The comps file for more information on this file, and how to use it.

Two other important files in the `base` directory are `hdlist` and `hdlist2` containing most of the header fields from all the RPMs in the `RPMS` directory. This means that all the interdependencies among RPM packages can be determined just by reading these files without having to read all the RPM packages which is quite convenient especially during FTP installs. Another use of these files is mapping package names to file names (eg. *perl* to *perl−5.004−6.i386.rpm*). This means that if you want to incorporate updates from RedHat (see section Including the updates) or add your own packages to the `RPMS` directory, you need to update `hdlist` and `hdlist2`. This is described later in Rebuilding the installer. Besides these files, the images from which the installation environment (i.e. kernel, python interpreter, anaconda, etc.) is loaded are found.

2.3. The "updates" directory

The `/pub/redhat/linux/updates` directory has updates for all releases of RedHat's distribution since version 3.0.3. This is the place to find software packages that have been updated for some reason or other. You should especially be aware of security updates. These are publicised on RedHat's errata page whenever a fix is available. The most important files found in the `updates` directory are:

```
drwxrwsr-x    3 root      root          4096 Jul 13 10:13 5.2
drwxrwsr-x    3 root      root          4096 Jul 13 10:13 6.0
drwxrwsr-x    3 root      root          4096 Jul 13 10:13 6.1
drwxrwsr-x    4 root      root          4096 Jul 13 10:14 6.2
drwxrwsr-x    4 root      root          4096 Jul 13 10:14 7.0
drwxrwsr-x    4 root      root          4096 Jul 13 10:14 7.1
drwxrwsr-x    4 root      root          4096 Jul 13 10:13 7.2
drwxrwsr-x    3 root      root          4096 Jul 13 10:14 7.3
drwxrwsr-x    3 root      root          4096 Jul 13 10:14 8.0
drwxrwsr-x    3 root      root          4096 Jul 13 10:14 9
```

The structure of each of these directories is similar to that described in the section The Redhat 9 tree. So you will find for each version, in the subdirectory `en/os/` a series of subdirectories representing the various architectures and a `noarch` and `SRPMS` subdirectories, for packages which work on every architecture or are in source form respectively.

```
drwxrwsr-x    2 root      root          4096 Sep 23 05:28 SRPMS
drwxrwsr-x    2 root      root          4096 Aug 28 18:25 athlon
drwxrwsr-x    2 root      root          8192 Sep 23 05:28 i386
drwxrwsr-x    2 root      root          4096 Jul 13 10:14 i486
drwxrwsr-x    2 root      root          4096 Aug 28 18:26 i586
drwxrwsr-x    2 root      root          4096 Aug 28 18:26 i686
drwxrwsr-x    2 root      root          4096 Jul 13 10:14 noarch
```

2.4. Differences for the 8.0 tree

The 8.0 distribution layout is almost identical to the one just described. The only major differences, in this respect, can be found in the images directory.

The `images` directory contains boot and drivers floppy images that can be copied to a diskette if needed. In the 8.0 release, there are three boot disk images available. The first boot image is called `boot.img`, and is required when installation is performed directly from a CD−ROM. If installing from a NFS mounted disk or FTP is required, the `bootnet.img` disk image is needed. Installs through PCMCIA adapters need the `pcmcia.img` floppy. See section Installation and references therein for details and consult the README file in the directory for a more detailed explanation of the various files.

2.5. Differences for the 7.x tree

The two distributions are fairly similar in this respect. The only changes which are of some interest to us (and easy to notice with a simple inspection of the main distribution tree) are represented by a missing `isolinux` directory and some changes in the `RedHat/base` directory. The first one is due to the way the installation CDs are made bootable in releases prior to 8.0 ("floppy emulation" has been superseded by "no emulation" in release 8.0), while the second is an effect of the migration of the `comps` file format to *XML* in Redhat releases after 8.0 (that's why it was renamed `comps.xml`). The `Redhat/base/comps` file is, in fact, a simple textual file with a quite inflexible syntax in releases prior to and including Redhat 7.3.

2.6. Differences for the 6.x tree

For release 6.2 (pub/redhat/linux/6.2/en/os/), the last of the 6 series, the organization is the following (the previous releases are mostly similar if not really equal, in this respect):

```
alpha/   i386/   sparc/
```

While the root of the i386 directory looks like this:

```
-rw-r--r--    1 root     root        18385 Sep  7  1999 COPYING
-rw-r--r--    1 root     root         3400 Mar  8  2000 README
-rw-r--r--    1 root     root        16300 Mar  8  2000 RELEASE-NOTES
-rw-r--r--    1 root     root         1908 Sep 25  1999 RPM-GPG-KEY
drwxr-xr-x    1 root     root          512 Sep 27 15:22 RedHat
drwxr-xr-x    1 root     root        17408 Sep 27 15:22 SRPMS
-rwxr-xr-x    1 root     root          538 Sep 26  1999 autorun
-rwxr--r--    1 root     root         2048 Mar  9  2000 boot.cat
drwxr-xr-x    1 root     root          512 Sep 27 15:22 doc
drwxr-xr-x    1 root     root          512 Sep 27 15:22 dosutils
drwxr-xr-x    1 root     root          512 Sep 27 15:22 images
drwxr-xr-x    1 root     root          512 Sep 27 15:22 misc
```

In the following paragraphs I will only list differences from the newest releases, what is not explicitly mentioned is (or is believed to be) unchanged.

The `doc` directory contains an abundance of information. Most importantly, the RedHat installation manual can be found in HTML format in the directory or on the Redhat website (Redhat 6.2 Installation guide). Next, there are the reference guide and the getting started guide. The documentation for the 7.x/8.0/9 releases is on a separate CD (in a different tree, on the ftp site).

The `images` directory contains boot floppy images that can be copied to a diskette if needed, like for 8.0, 7.3 and 9. See section Installation and references therein for details. The `misc` directory contains source and executables of a number of programs needed for the installation.

The most important part of the directory tree is (again) rooted in the `RedHat` directory:

```
drwxr-xr-x   2 root     root    28672   Oct 26 09:01   RPMS
drwxr-xr-x   2 root     root     4096   Oct 26 09:01   base
-rw-r--r--   1 root     root        0   Jan 19  1999   i386
drwxr-xr-x   6 root     root     4096   Oct 26 09:01   instimage
```

The `RPMS` directory should be already known to you. See the section RPM packages for more informations. The `base` directory holds different book−keeping files needed during the installation process, like for releases 7.3, 8.0 and 9. The only noticeable differences being represented by a single `hdlist` file and a missing `stage2.img` file whose functionalities should be provided by the files included in the `instimage` directory. This contains, in fact, a bare−bones live file system with a number of programs and shared libraries needed during the installation procedure.

The `updates` directory is really similar to the one described for release 9 with the only difference of having more architecture related directories.

3. RPM packages

The major part of the Red Hat distribution consists of a set of RPM (Redhat Package Manager) files. An RPM package typically contains binary executables, along with relevant configuration files and documentation. The rpm program is a powerful package manager, which can be used to install, query, verify, update, erase and build software packages in the RPM format. `Rpm` convieniently maintains a database of all the software packages it has installed, so information on the installed software is available at any time.

The binary RPM files in the distribution have been built on a system running the distribution itself. This is important, because most of the programs in the packages rely on shared libraries. From RedHat version 5.0, the new version 2 of the GNU standard C library (which is 64–bit clean) has been used. This version of the library is commonly referred to as *glibc* or in Linux: *libc 6*. All executables in the distribution have been linked against this library. If you attempt to install binary files from a different distribution, chances are that they will not work, unless you install the libc5 package for backwards compability. There are also incompatibilities between the various version of the Redhat Package Manager itself which will make the installation of some packages fail even on machines they are supposed to (and they would probably) run on.

The names of the RPM packages contain the suffix *.arch.rpm*, where *arch* is the architecture, usually having the value *i386* for Intel platform binaries. The packages you install must match the versions of the shared libraries available on the machine. The rpm program is usually quite good at ensuring that this is indeed the case, however, there are ways around this check, and you should be sure that you know what you are doing if you force installation of packages this way. However, using the RedHat installation boot disk, it is ensured that the correct set of RPM packages are installed on the machine.

If you discover a RPM package that was not installed on your system during the installation process, don't despair. At any time, you may (as root) install RPM packages, for example:

```
# rpm --install  WindowMaker-0.18-1b.i386.rpm
```

You can even install directly from the Internet, if you know the URL of a RPM package:

```
# rpm --install ftp://rufus.w3.org/redhat-contrib/noarch/mirror-2.9-2.noarch.rpm
```

If you want to update (or install if it's not present on the machine) a RPM package, use the command:

```
# rpm --update  WindowMaker-0.18-1b.i386.rpm
```

If you want to update a RPM package only if a previous version is already installed on the machine use the command:

```
# rpm --freshen  WindowMaker-0.18-1b.i386.rpm
```

Another version of the RPM packages contains the original sources used to build the binaries. These packages have the suffix *.src.rpm* and are situated in the `SRPMS` directory. These packages compose the last two CDs and part of the third one out of the five which compose the 8.0 (or 7.3) release. For release 9 they are on three separate CDs. For 6.2 (and previous, not too old, versions), things change a bit because there is only one installation CD not comprising the SRPMS packages, which you can burn on a different disc, if you want.

To obtain more informations on the Redhat package manager, I suggest you to read the man pages and the fairly detailed book maximum rpm.

In the next section, I will introduce a C program which will be used in various scripts throughout the rest of the howto. It just returns, given two versions of the same RPM package, which one is newer. This program is based on the code used in the Redhat Package Manager (release 4.1) and is used when the *−−freshen* option is given.

3.1. Comparing two versions of a RPM package

The C code included in the three files Makefile, rvc.h, rvc.c was extracted from the Redhat Package Manager and (slightly) modified to fit our needs. They form a simple C program which given two versions A and B of a package returns 1, 0 or −1 if A is respectively newer, equal or older than B and other values in case of error (you can read the code comments for more detailed informations). To compile the program (you need the `make` program and the `gcc` C compiler), put the files in the same directory and issue the command:

```
        $ make
```

This program is needed by almost every script used in the following sections and is found setting the *RVC* variable in the file rhcd.conf.

You can find a copy of the source and a precompiled version in the archive rhcd−sripts.tar.gz in the `rpmvc` directory.

There was an error in the way this program was used by the *updateDist.sh (ver. < 1.17)* and *updateCD.sh (ver. < 1.12)* scripts. I strongly suggest to avoid versions of the scripts which have lower release numbers than the reported ones, even if the problem doesn't show up really frequently (at least apparently).

4. Obtaining your local copy of the distribution

You need a copy of the distribution on a writable disk which is accessible from the computer having the CD writer (duh!). If you want to incorporate the latest updates, this directory should (also) be accessible from a Linux machine, either from a local disk, an NFS mounted disk on a different computer, or a JAZ disk. You could copy the distribution from the RedHat CDs (recommended), or you could get it via FTP. If you choose to use FTP, there are two ways of doing it. You can use the *wget* based shell script presented in the following section or the *mirror* package as suggested in versions up to and including 1.34 of the howto (reported in section Using mirror).

4.1. Using wget and bash

This is not the simplest, even if, probably, the most accurate way. I like it because it works comparing the RPM versions of the files and not the dates/times or names (like the standard mirroring packages) and it checks the signatures of the updates each time it downloads some of them if configured to do so by means of the *CHECKSIG* variable in the rhcd.conf file.

Create a directory to hold the installation files and *cd* into it, then issue the command (which will download ~3Gb of data on your hard drive):

```
        $ wget -r -c -t0 -l0 --retr-symlinks -nH --cut-dirs=9 \
            ftp://ftp.mirror.ac.uk/sites/ftp.redhat.com/pub/redhat/linux/updates/7.3/en/os/i386
```

You will probably want to change the ftp download mirror and, consequently, the parameter passed to the *−−cut−dirs* option. That's used, in fact, together with *−nH* to avoid the recreation of the ftp site directory hierarchy. For more information on how to use the option correctly you can have a look at the wget documentation and man page.

If you want to exclude one or more directories from the download, you can use the *−X list* option, where *list* represents a comma−separated list of directories. For example to exclude the `SRPMS` directory from the previous download, you would use:

```
        $ wget -r -c -t0 -l0 --retr-symlinks -nH --cut-dirs=9 \
            -X /sites/ftp.redhat.com/pub/redhat/linux/updates/7.3/en/os/i386/SRPMS \
            ftp://ftp.mirror.ac.uk/sites/ftp.redhat.com/pub/redhat/linux/updates/7.3/en/os/i386
```

This could be useful if you consider the size of the `SRPMS` directory (~1.2GB), or at least, I find it useful.

If you want to check the GPG signatures to make sure of the authenticity of the packages (which is something I suggest) you should install the *gnupg* package (needed only on Redhat 7.3) and import the *security@redhat.com* public key you can find in the root directory of the CDs (`RPM−GPG−KEY`) or on the RedHat website. The key is imported by running the command: *gpg −−import <filename>* in releases up to and including 7.3, which is to be changed to read *rpm −−import <filename>* for releases 8.0 and 9 (for more informations on this have a look at the GNU Privacy Guard and at the RPM − Redhat Package Manager websites).

If you want to check the rpm packages you can do it using the following command (I'm assuming you are issuing it from the directory you have completed the downloads in):

For releases up to and including 7.3:

```
$ find . -name "*.rpm" -exec rpm -K --nopgp {} \; |grep "NOT *OK"
```

For release 8.0 and 9 (and for future releases as well I guess):

```
$ find . -name "*.rpm" -exec rpm -K {} \; |grep "NOT *OK"
```

If you don't want to "bother" yourself with all these steps, I hope you want to check (at least) for the integrity of the downloaded files (which doesn't mean nobody has tampered with them), verifying the md5 signatures. This is done with:

For releases up to and including 7.3:

```
$ find . -name "*.rpm" -exec rpm -K --nopgp --nogpg {} \; |grep "NOT *OK"
```

For release 8.0 and 9 (and for future releases as well, I guess):

```
$ find . -name "*.rpm" -exec rpm -K --nosignature {} \; |grep "NOT *OK"
```

The content of a Red Hat distribution does not change between releases, so you only need to download these packages *ONCE*. All changes to the distribution are in the `updates` directory. Thus, if you want to keep an up−to−date mirror of the Red Hat distribution, you only need to keep the `updates` directory current. This is done using the script updateDist.sh. Before using this script you have to configure the rhcd.conf configuration file and export a *RHCDPATH* variable pointing to the directory where this file is.

```
$ export RHCDPATH=/home/luigi/tmp/rhcd-scripts
$ sh updateDist.sh
```

The script will download the new updates excluding the subdirectories contained in the *EXCLUDELIST* variable, moving the old ones (i.e. just superseded by new versions) to the directory represented by the *OLDDIR* variable after having completed two tests. The first test compares the `.listing` files generated by `wget` to the content of the local directories to make sure all the files were downloaded. The second test verifies the packages signatures depending on the values of the two variables *CHECKSIG* and *USEGPG* (set both of them to "yes" if you want the operation to be completed). In case of a failure in the signature checking process the script will move the offending packages to *OLDDIR* assigning them the ".UPDcheckfail" extension and exit without moving the old updates to *OLDDIR*.

4.2. Using mirror

Mirror is a sophisticated perl script that compares the content of a directory on a remote site with a local directory. It will use FTP to fetch the files that are on the remote site but not the local site, and delete files on the local site that are not on the remote site. The mirror program is configured with a configuration file. The mirror package is available as an RPM from rufus.w3.org. Make your local copy `mirror.redhat` of the mirror configuration file, and edit the relevant fields at the top of the file. After the default section, define these packages:

```
package=updates
  site=ftp.mirror.ac.uk
  exclude_patt=(SRPMS/)
  remote_dir=/sites/ftp.redhat.com/pub/redhat/linux/updates/7.3/en/os/i386
  local_dir=/home/luigi/tmp/redhat-cd/redhat-7.3-updates

package=dist
  site=ftp.mirror.ac.uk
  exclude_patt=(SRPMS/)
  remote_dir=/sites/ftp.redhat.com/pub/redhat/linux/7.3/en/os/i386
  local_dir=/home/luigi/tmp/redhat-cd/redhat-7.3
```

The following command will download a copy of the entire RedHat tree on your local disk. ***Think*** before you do this, you are about to transfer approximately 1.5Gb of data (if you have excluded the `SRPMS` directory)!

```
$ mirror -pdist mirror.redhat
```

This will mirror the Red Hat FTP site on your local disk. The content of a Red Hat distribution does not change between releases, so you only need to download this package *ONCE*. All changes to the distribution are in the `updates` directory. Thus, if you want to keep an up–to–date mirror of the Red Hat distribution, you only need to keep the `updates` directory current. This is done using the command

```
$ mirror -pupdates mirror.redhat
```

You can run this regularly, say, once a week, through a cron script. The RedHat distribution is available on a great number of FTP servers around the world, which are updated daily from the master site at ftp.redhat.com. You should choose an FTP site close to you, see the RedHat list of mirror sites.

I haven't personally tested this procedure. It was the only proposed one for the older versions of the howto (up to version 1.34, regarding RedHat <=6.1).

5. Including the updates

There are three steps involved, the first two are (almost) equal in all the releases, while the last one changes quite a bit because of the changes in the anaconda installer:

i. Correct the file protection modes
ii. Replace updated RPMs
iii. Rebuild the installer

To incorporate the updates, you need write access to the distribution directory from a Linux machine, with a working version of rpm installed, while *to rebuild the anaconda installer you need to use a release of Redhat Linux equal to the one you are rebuilding the installer for (otherwise the procedure will fail)*. If you maintain a mirror of the `updates` directory, you can at any time produce a CD including the current updates by repeating these steps.

5.1. Correcting the file protection modes

During the installation process of the releases up to and including 6.2, some programs are run directly off the CD. Unfortunately, the FTP program does not always preserve the protection modes of the files and directories that are copied. Therefore, it is necessary to make sure that execute permission is given to programs, shell scripts and shared libraries, before the directory is burned on the CD. This is done by running the updatePerm.sh script on your local copy of the distribution. It is really needed only for version 6.2 and older, the only part useful to the 7.3/8.0/9 releases procedure is the directories permissions update, even if the rest won't hurt and things are kept coherent. It is almost equal to the `updatePerm` script included in the previous version of the howto, just some slight changes were made. Before using this script you have to configure the rhcd.conf configuration file and export a *RHCDPATH* variable pointing to the directory where this file is.

```
$ export RHCDPATH=/home/luigi/tmp/rhcd-scripts
$ sh updatePerm.sh
```

5.2. Replacing the updated RPMS

The updateCD.sh script copies all the new files from the update directory to the RPMS (and SRPMS) directory. The script uses the `rvc` program which was presented in section Comparing RPM versions to determine which packages in the updates directory are more recent. Older packages are moved to the *${OLDDIR}* directory. If the *CHECKSIG* variable is set to "yes", all the packages in the main tree will have their signature checked for correctness. If a package fails the signature check (the kind of check is configured by means of the *USEGPG* variable whose value is assigned in the file rhcd.conf), it is moved to the *OLDDIR* directory with an added extension of "CDcheckfail".

Before using this script, you have to configure the rhcd.conf configuration file and export a *RHCDPATH* variable pointing to the directory where this file is.

```
$ export RHCDPATH=/home/luigi/tmp/rhcd-scripts
$ sh updateCD.sh
```

After having incorporated the updates in the main `RedHat/RPMS` directory, your copy of the distribution is no longer a mirror of the Red Hat distribution site. Actually, it is more up−to−date! Therefore, if you attempt to mirror the distribution using mirror, older versions of the RPM's that have been updated will be downloaded once more, and the updates deleted. The bash/wget based procedure doesn't suffer from the problem, but will leave the main tree in an incoherent state. Old and new packages will be in this case mixed together, but you can find and remove them wrapping the `rvc` binary in a simple shell script (which I will leave as an exercise for the reader...).

5.3. Rebuilding the installer

Things have changed pretty much in this section with the introduction of the anaconda installer (as of release 6.1) and with the considerable increment in size (and ... number of CDs) the 7.x/8.0 distributions have seen. Until release 6.2, the only step composing this section was represented by generating a new `hdlist` file. With release 6.2, this appears to be true only to a certain extent, because of the changes in the anaconda installer, in the *rpm* software itself (from version 3.x to 4.x) and the migration of the updated packages to this new version (updates for release 6.2 are in fact packaged with both major releases of the `rpm` software). We will consider three different procedures trying to cover all the releases.

5.3.1. RedHat <= 6.1

5.3.1.1. Regenerating the hdlist file

When installing from the CD, the installation program on the CD relies on the file `RedHat/base/hdlist` describing what RPM packages are available on the CD. The `hdlist` file can be generated by the program `misc/src/install/genhdlist`. This program must be run with the absolute path to the root of the distribution as the only argument. Here is the `updateHdlist` script which calls that program (from version 1.34 of this howto):

```
#!/bin/bash

RHVERSION=6.1
ARCH=i386

echo generating hdlist...
RHROOT=/home/luigi/tmp/redhat-${RHVERSION}
GENHDDIR=${RHROOT}/${ARCH}/misc/src/anaconda/utils

chmod u+x ${GENHDDIR}/genhdlist
chmod 644 ${RHROOT}/${ARCH}/RedHat/base/hdlist
${GENHDDIR}/genhdlist ${RHROOT}/${ARCH} || echo "*** GENHDLIST FAILED ***"

exit 0
```

Important note for RedHat < 6.1

The installation in RedHat 6.1 is completely changed from earlier versions, and RedHat has introduced *anaconda*. The `genhdlist` program is now found in a different place, so in the script above, we used

```
GENHDDIR=${RHROOT}/${ARCH}/misc/src/anaconda/utils
```

while for releases up to (and including) 6.0 that line should read

```
        GENHDDIR=${RHROOT}/${ARCH}/misc/src/install
```

In some cases, `genhdlist` fails to run, because the executable is not statically linked. In such a case, you can add a new line `${RHROOT}/${ARCH}/RedHat/instimage/usr/lib` in `/etc/ld.so.conf` and run `ldconfig -v`.

Another solution is to recompile `genhdlist`. The following modification to the `updateHdlist` script worked under RedHat 5.2:

```
        #!/bin/bash

        RHVERSION=6.1
        ARCH=i386

        RHROOT=/misc/redhat/redhat-${RHVERSION}
        GENHDDIR=${RHROOT}/${ARCH}/misc/src/anaconda/utils

        echo Compiling genhdlist...
        sed -e 's/FD_t/int/' \
            -e 's/fdOpen/open/' \
            -e 's/fdClose/close/' \
            -e 's/fdFileno//' < ${GENHDDIR}/genhdlist.c > /tmp/genhdlist.c
        cc -o /tmp/genhdlist -I/usr/include/rpm /tmp/genhdlist.c -lrpm -lz

        echo generating hdlist...
        chmod 644 ${RHROOT}/${ARCH}/RedHat/base/hdlist
        /tmp/genhdlist ${RHROOT}/${ARCH} || echo "*** GENHDLIST FAILED ***"

        exit 0
```

In this version of the script, a copy of the C source of `genhdlist.c` is piped through `sed` to create a copy in `/tmp` that will compile under RedHat 5.2. This version of `genhdlist` is then used to create the `hdlist` file

Important note for RedHat 5.2

As distributed with RedHat version 5.2 and earlier, `genhdlist` CRASHES if there are files in the `RedHat/RPMS` directory which are *not* RPM files! This causes problems, because in the 5.2 distribution, there are a couple of non−RPM files named `ls-lR` and `ls-lR.gz` in `RedHat/RPMS`. Therefore, you must remove all non−RPM files from the directory. Alternatively, you can apply the patch genhdlist.c.diff to `misc/src/install/genhdlist.c` and do a *make*. The patch will cause `genhdlist` to ignore any non−RPM files.

5.3.1.2. Creating the CD iso image

You'll need to create an image file which will be written to the CD. This file will be 500Mb or more so find a partition with enough free space. You may need to be root to use mount and cdrecord. Here you will prepare the iso image of the bootable CD to be burned. It is actually, not strictly necessary, to create a bootable CD, because you could use a boot floppy instead of it, but it's definitely a nifty feature (and makes your disc more similar in behaviour to the original one). These are the commands I use to complete the task:

```
$ mkdir /images-destination-dir
$ mkisofs  -r  -J  -T  -v  -V "Red Hat 6.1 (Hedwig)" \
    -c boot.cat  -b images/boot.img \
    -o /images-destination-dir/i386-disc.iso .
```

This is needed to burn the (bootable) disc and is executed from the top level directory of the distribution. The `/images-destination-dir` directory is the container for the iso image you are generating, and it must exist (obviously) before starting the procedure. In the following table you can read a brief explanation of the various options and their meanings (most of it was extracted from the `mkisofs` man page).

Table 1. mkisofs options and parameters

−r	Rock Ridge extensions with useful values for the permission bits
−J	Joliet extensions to use the cd with some different operating systems
−T	Generate a TRANS.TBL file in each directory to map correctly the file names even on systems which do not support the Rock Ridge extensions.
−v	be verbose
`-V <volid>`	Specifies the volume ID (volume name or label) to be written into the master block.
`-c <boot catalog>`	Specifies the path and filename of the boot catalog to be used when making an "El Torito" bootable CD. The pathname must be relative to the source path specified to mkisofs.
`-b <eltorito boot image>`	Specifies the path and filename of the boot image to be used when making an "El Torito" bootable CD. The pathname must be relative to the source path specified to mkisofs and specify a floppy image (which is why we use one of the floppy images found on the original CD. You may want to change this with the `pcmcia.img` image to install using pcmcia devices like network cards or CDROM readers.
`-o <filename>`	Name of the file containing the generated iso image
.	This is the root directory for our generated iso image (we are working from the root directory of every CD, so a dot is enough).

You will find details of how to burn the image on a media in Burning the CD. The `mkisofs` and `cdrecord` steps can be executed by means of a graphical frontend like X−CD−Roast which should currently support the creation of bootable CDs (I've never used it, so don't expect me to give you any explanation).

5.3.2. RedHat 6.2

Apparently, this is a problem child when it comes to burning an uptodate CD. The introduction of version 4 of the Redhat Package Manager (RPM), made the procedure to update the anaconda installer fail. So the procedures listed will work only if the updated packages were built using a version of the RPM software which is older than or equal to 3.0.4 (so, basically, 3.0.4 or 3.0.5).

If you are using the original packages from Redhat, you have to avoid using updates released after 28 March 2001 (which is a bit useless, in my opinion) or you have to rebuild the packages using the old rpm format. Details on the downgrade procedure and tools which implement it can be found in the document rpmhack. I have not personally tested this procedure, even if it appears to work if you read about it on the anaconda−devel and kickstart mailing lists (you can find them on the mailing lists section of the Redhat website.

If you decide to stick to the old original packages and complete the update (using the rpm 4.0.2 packages after the installation is finished) there are two possible ways of doing it, depending on which kind of updates you want to complete the CD with. If some of the updates regard directly the installation process (e.g. kernel, python, kudzu), you will have to use the installer rebuilding procedure explained in the document Building a Red Hat Linux 6.2 CDROM, otherwise you can still use the old procedure (the one for releases previous to and including 6.1 explained in the previous section). The last two steps, which are, respectively, creating the iso image and burning the actual media are described in Creating the CD iso image and Burning the CD, respectively.

5.3.3. RedHat 9, 8.0 and 7.3

Once again a lot of things have changed with the release of the 7.x series of the distribution. There are now more operations to complete to obtain a fresh and uptodate series of CDs. Exactly, they have become more than one with release 7.0 and now the tree has to be split to fit on the media. This is done by means of the `splitdistro` script, which is written in python like most of the anaconda installer. To complete this part, you *must* use a Linux RedHat 7.3, 8.0 or 9 machine with the *anaconda−runtime* package installed (it will probably have version 7.3.7, 8.0.4 or 9.0.4), depending on the release you want to rebuild. The procedure is composed by seven steps:

i. Regenerating the `hdlist` and `hdlist2` files
ii. Updating the `comps.xml` (or `comps`) file
iii. Rebuilding the installer
iv. Splitting the distribution in CD−sized chunks
v. Regenerating the `hdlist` and `hdlist2` files (again)
vi. Generating the iso images
vii. adding and checking the md5 signatures in the iso images

All the steps are grouped together in a single script presented in the last section.

5.3.3.1. Preliminary operations on the main tree

Some of the scripts included in the `anaconda-runtime` package need the main tree to be moved in a subdirectory named like the architecture we are building for (so `i386/` for me). We will move everything to such directory before starting the procedure and change the invocation of the scripts which don't need this modification.

For redhat 9 and 8.0:

```
$ chmod -R u+w /absolute-path-to-toplevel-dir
$ mkdir -p /absolute-path-to-toplevel-dir/i386
$ cd /absolute-path-to-toplevel-dir
$ /bin/mv * i386
```

You should change "/absolute−path−to−toplevel−dir" with the *absolute path* of the directory where the root of your local copy of the distribution is located (maybe somewhere on some hard drive). You will get an error, from the execution of the last command, because the `i386/` directory cannot be moved under itself, but you don't need to worry about that.

For redhat 7.3:

```
$ chmod  -R  u+w  /absolute-path-to-toplevel-dir
$ mkdir  -p  /absolute-path-to-toplevel-dir/i386
$ cd /absolute-path-to-toplevel-dir
$ for i in `ls` ; do [ $i != "SRPMS" -a $i != i386 ] && /bin/mv $i i386 ; done
```

You shouldn't receive any error message this time from the last command (hopefully).

5.3.3.2. Regenerating the hdlist and hdlist2 files

This is done by means of the following two commands with the help of the `genhdlist` program.

```
$ /usr/lib/anaconda-runtime/genhdlist  /absolute-path-to-toplevel-dir/i386
$ chmod  644 /absolute-path-to-toplevel-dir/i386/RedHat/base/hdlist{,2}
```

Once again, "/absolute−path−to−toplevel−dir" is the *absolute path* of the directory where the root of your local copy of the distribution is located. The second command is needed to make sure the correct permissions are set for the file. You should already have an idea of what these files are about if you have read through The Redhat directory.

5.3.3.3. Update the *comps.xml* file

In Redhat Linux 8.0 the format of the comps file has completely changed and it's now based on XML. It provides much more flexibility and ease of customization as you can read in The comps file. If you have modified or intend to modify the list of the installed packages, you need to complete this step. This, in turn, implies having the modified version of comps−9.tar.gz comps−9.tar.gz (the original one doesn't work for me) or comps−8.0.tar.gz package (depending on the release you are building) including the master comps file found on the Redhat website and the `comps-extras` rpm package installed. Follow these steps for Redhat 9 and 8.0:

```
$ cd /some-dir-of-your-choice
$ tar xzvf /path-to-comps-9.tar.gz/comps-9.tar.gz
$ cd comps
$ make
$ cat comps-milan.xml |sed 's!</comps>!!g' >comps-tmp.xml
$ /usr/share/comps-extras/getfullcomps.py  comps.xml \
    /absolute-path-to-toplevel-dir i386 >> comps-tmp.xml
$ echo '</comps>' >> comps-tmp.xml
$ cp comps-tmp.xml /absolute-path-to-toplevel-dir/i386/RedHat/base/comps.xml
```

Beside "/absolute−path−to−toplevel−dir", you should take care of assigning valid names to "/some−dir−of−your−choice" and "/path−to−comps−9.tar.gz". The rest of the commands can be just copied. And... you must (obviously) change 9 to read 8.0 if you are building a Redhat 8.0 distribution.

Again, before issuing the *make* command, you should modify the file `comps-milan.xml.in` using your favourite text editor and following the guidelines and the suggestions found in The comps file and on the anaconda comps section of the Redhat website.

The script presented in the last section will execute all the steps needed after the *make* command, using the *COMPSFILE* variable to find the `comps-milan.xml` file (it doesn't need to have that name, I'm just using the original name, but you can change it if you want).

If you are using Redhat 7.3, the `comps` file (have you noticed the different name...) is a textual file with a completely different syntax described in some more detail in The comps file. In this case, the only necessary operations are modifying the file to suit your needs and copying it to the `RedHat/base/comps` file in the main tree overwriting the original one.

5.3.3.4. Rebuilding the installer

This will rebuild the anaconda installer in your local copy of the distribution using your updated packages. For Redhat 9 execute:

```
$ /usr/lib/anaconda-runtime/buildinstall  \
    --pkgorder /absolute-path-to-toplevel-dir/pkgorder.txt  \
    --comp dist-9 --product "Red Hat Linux" --version 9  \
    --release "Redhat 9 (Shrike)" /absolute-path-to-toplevel-dir/i386
```

Where, once again, "/absolute–path–to–toplevel–dir" is the directory where the root of your local copy of the distribution is located.

For Redhat 8.0, the procedure is pretty much the same (the "––product" option is missing):

```
$ /usr/lib/anaconda-runtime/buildinstall  \
    --pkgorder /absolute-path-to-toplevel-dir/pkgorder.txt  \
    --comp dist-8.0 --version 8.0  --release "Redhat 8.0 (Psyche)" \
    /absolute-path-to-toplevel-dir/i386
```

Or if you are still using Redhat 7.3 (as I am):

```
$ /usr/lib/anaconda-runtime/buildinstall  \
    --pkgorder /absolute-path-to-toplevel-dir/pkgorder.txt  \
    --comp dist-7.3 --version 7.3 /absolute-path-to-toplevel-dir/i386
```

The absence of the (mandatory in 8.0) ––*release* option is the only noticeable difference.

5.3.3.5. Split the distribution

This will create five directories, each one corresponding to a different CD and will put in them hard links to the real files contained in your local copy of the distribution.

This will not work at all for Redhat 7.3 if you don't use the modified version of the `splitdistro` script reported in the next paragraph. For Redhat 8.0 and 9, a modified version of `splitdistro` is provided mainly because even if the problems in the previous script were fixed, the execution now fails if there are not enough packages to fill all the CDs (the first four completely and the last one even just partly).

```
$ $/usr/lib/anaconda-runtime/splitdistro  \
  --fileorder /absolute-path-to-toplevel-dir/pkgorder.txt  --release \
    "Redhat 9 (Shrike)"  /absolute-path-to-toplevel-dir  i386
```

The only thing you need to change for 8.0 and 7.3 is the string passed to the *−−release* option (which should read "Redhat 8.0 (Psyche)" or "Redhat 7.3 (Valhalla)")

For Redhat 7.3 the version of the splitdistro7.3 (python) script used was extracted from the *anaconda−runtime 7.3.7* package and modified by me. You should sustitute it to the original one (maybe after copying the latter) named `/usr/lib/anaconda−runtime/splitdistro`.

The only modification (apart from some small fixes), the script went through, is a change in its behaviour if the `SRPMS` directory is not found (doesn't terminate, but generate the CDs without source packages).

For Redhat 8.0 the version of the splitdistro8.0 (python) script used was extracted from the *anaconda−runtime 8.0.4* package and modified once again by me to obtain some improvements I felt the need for. You should sustitute it to the original one (maybe after copying the latter somewhere) named `/usr/lib/anaconda−runtime/splitdistro`. Anyway, the original one works well, if you want to build a distribution which has all the *SRPMS* packages (so to fill all the 5 CDs otherwise the script will fail).

The only modification the script went through is a change in its behaviour if the `SRPMS` directory is not found (doesn't terminate failing, but generates the CDs without source packages) or there is one CD which hasn't any package on it (instead of failing, generates an empty directory).

For Redhat 9 you can find a copy of the script with the same modifications applied to the version included in release 8.0 here: splitdistro9. Everything said for Redhat 8.0 in the previous paragraph applies to release 9.

5.3.3.6. Regenerating the hdlist and hdlist2 files

This is needed to recreate the hdlist and hdlist2 files, using some of the informations obtained in the previous steps. There are no differences between 7.3, 8.0 and 9 for this execution of the program The command to issue is the following:

```
$ /usr/lib/anaconda-runtime/genhdlist  \
    --fileorder /absolute-path-to-toplevel-dir/pkgorder.txt  --withnumbers \
    /absolute-path-to-toplevel-dir/i386-disc[1-3]
```

As you can see, there are two new options passed to the program, if you remember the first run of it. The first one, *−−fileorder*, tells genhdlist to use the file `pkgorder.txt` we generated in the second step (rebuild the installer). This file keeps informations on how the packages were split on the different CDs and is used by the installer to determine in which order the packages should be installed. Basically, if you avoid using it, you will (probably) end up swapping the various CDs many times during the installation. The *−−withnumbers* option is needed to associate a CD number to every package (as you can see, a wildcard indicating the first 3 iso images is used).

5.3.3.7. Generating the iso images

Here you will prepare the iso images to be burned on the actual CDs. There are two different commands to be used for the first disc and for the rest of them. This is due to the need of obtaining a first CD which is bootable. This is actually, not strictly necessary, because you could use a boot floppy instead of it, but it's definitely a nifty feature (and makes your discs more similar in behaviour to the original ones). These are the commands I use to complete the task:

```
$ mkdir /images-destination-dir
$ mkisofs  -r  -J  -T  -v  -V "Red Hat 9 (Shrike) disc 1" \
    -c isolinux/boot.cat  -b isolinux/isolinux.bin -no-emul-boot \
    -boot-load-size 4 -boot-info-table -o /images-destination-dir/i386-disc1.iso .
```

This is needed to burn the first (bootable) disc for RedHat 8.0 and 9 (with no floppy emulation) and is executed from the top level directory of the distribution. The `/images-destination-dir` directory is the container for the five iso images you are generating, and it must exist before starting the procedure. The only thing which needs to be changed for Redhat 8.0 is the volume name (it should be "Red Hat 8.0 (Psyche) disc 1").

```
$ mkdir /images-destination-dir
$ mkisofs  -r  -J  -T  -v  -V "Red Hat 7.3 (Valhalla) disc 1" \
    -c boot.cat  -b dosutils/autoboot/boot.img \
    -o /images-destination-dir/i386-disc1.iso .
```

This is needed to burn the first (bootable) disc on 7.3 and is executed from the top level directory of the distribution (this time with floppy emulation).

The rest of the images can be written by means of this "for" loop

```
$  for i in `echo 2 3 4 5` ; do mkisofs  -r  -J  -T  -v  \
     -V "Red Hat 9 (Shrike) disc ${i}"  \
     -o /images-destination-dir/i386-disc${i}.iso . ; done
```

The loop just presented will prepare the last four images giving them the correct numbers. As you can see, there are just two missing options from the first run, and, as you can guess, they are needed only to create a bootable CD. In Creating the CD iso image, you can read a brief explanation of the various options and their meanings (most of it was extracted from the man page). Again if you are building a Redhat 8.0 you should change the volume name to read "Red Hat 8.0 (Psyche) disc ${i}".

5.3.3.8. Implant and check the md5 signatures in the iso images

This is actually an optional step but it permits the use of the "checkmedia" option to verify the CDs signatures before installing them, so to guarantee their correctness.

The following commands permit to inject and verify an md5 signature on an iso image:

```
$ /usr/lib/anaconda-runtime/implantisomd5 iso-image
$ /usr/lib/anaconda-runtime/checkisomd5 iso-image
```

After completing all these steps, we will find ourselves with the five CD images to burn. Considering that typing all this stuff is a bit time consuming, in the next section is presented a script, which will complete all of the listed operations in a single run (do not forget to configure the parameters properly).

5.3.3.9. Putting all the steps together

The updateBuild.sh script will execute all the steps needed to rebuild the distribution CDs for RedHat 7.3, 8.0 or 9 in a single run (as root). Before using this script you have to configure the rhcd.conf configuration file after exporting a *RHCDPATH* variable pointing to the directory where this file is. If you want to include a modified *comps.xml* (or *comps*) file in your CDs as explained in The comps file, you should copy it into the location defined by means of the *COMPSFILE* variable now (before executing the script). Don't forget to add the modified `splitdistro` script to the `/usr/lib/anaconda-runtime` directory if you need it.

```
# export RHCDPATH=/home/luigi/tmp/rhcd-scripts
# sh updateBuild.sh
```

6. Burning the CD(s)

This is composed by an optional and a required steps. Remember that, probably, you have to be "root" on your machine to run `cdrecord` .

6.1. Test the image(s)

If you're paranoid, you can test your new disk image(s) by mounting it. If you forgot to fix the file permissions or set the rock ridge extensions then the error will be obvious here since the file names and directory structure will be wrong. The (optional) test can be done by issuing the command:

```
# mount -t iso9660 -o ro,loop=/dev/loop0 iso-image /mnt/cdrom
```

Where "iso–image" is the name you gave to the iso image file to be mounted (which is the only one for releases up to and including 6.2). When you're done, don't forget to unmount it

```
# umount /mnt/cdrom
```

6.2. Burn the disk(s)

Be sure to set the correct parameters for your device. This command, for example, is for a 4X CDR, which is quite slow, by the way. Moreover, it is assumed that the CD writer is on SCSI bus 0, with ID number 0 and LUN 0 (you can obtain these values by issuing a *cdrecord −scanbus* and assign them to the *−dev*= parameter).

```
# cdrecord -v speed=4 dev=0,0,0 /images-destination-dir/disc1.img
```

7. The comps file

The comps file defines how the packages are bundled during the installation. In the Red Hat distribution, this is done according to the functionality they provide, for example:

- Printer Support
- X Window System
- GNOME
- KDE
- Mail/WWW/News Tools
- ...
- Kernel Development
- Extra Documentation

Sometime during the installation process, the user is presented with a dialog called "Components to install". Some of the components have been preselected, and others not. The last item on the components list is called "Everything". On the dialog box, there is also an option that enables the user to customize exactly what packages will be installed. Customizing the installation by hand, or selecting "Everything" in the components list is the only way to have your own packages installed unless you modify the RedHat/base/comps file.

7.1. Format of comps file in RedHat versions < 6.1

The comps file currently starts with a header describing the version of the comps format, followed by an empty line.

```
0.1
<empty line>
```

After this, the components are listed, separated by empty lines:

```
<component 1>
<empty line>
<component 2>
<empty line>
....
<component n>
<empty line>
EOF
```

Each component has the following definition:

```
(0|1) (--hide)? <name>
<RPM 1>
<RPM 2>
...
<RPM n>
end
```

Before the name of each component, 0 or 1 is given. A value of 1 here means that the component is chosen by default, and 0 means it's not. The option *−−hide* means that you will not see the entry, unless you choose "expert" installation. The first component is called "Base", and that is special, in the sense that it *must* be present and it does not show up in the dialog (you can't deselect the base installation, which makes sense...). Next follows a list of rpm packages belonging to that component. Note that this is the package name stored *in the rpm file*, and *not* any part of the file name of the package (although it should be the same by convention).

By adding your packages to the `comps` file, you can customize your own distribution, and make sure that your packages will be installed by default. One thing to be careful about is interdependence among your packages, but here, you are on your own :−) A word of warning: be careful not to add or remove extra whitespace in the file. Examine the existing `comps` file (make a copy of the original) to see how it's done (or check `i386/misc/src/install/pkgs.c` if you want to see how the file is parsed).

7.2. Format of comps file in RedHat version 6.1

With RedHat version 6.1, the format of the `comps` file has changed. The decoding takes place in `${RHROOT}/${ARCH}/misc/src/anaconda/comps.py`. I didn't analyze yet this python script and the following rules were obtained only by reading the file and testing some configurations for it.

In release 6.1, the definition of *component* is extended to include some more optional elements beside the *<RPM>* ones. These elements are:

```
          <arch-dependent-RPM 1>
          ...
          <arch-dependent-RPM n>
          <required-component 1>
          ...
          <required-component n>
          <component-dependent-RPM 1>
          ...
          <component-dependent-RPM n>
```

An <arch−dependent−RPM> defines a dependency between a package and specific architecture and has the following definition:

```
          (!)?arch: <RPM>
```

So it can, for example, present itself, in the real world, as:

```
          !alpha: kernelcfg
```

which means: if architecture is not alpha then install package *kernelcfg*.

Or as:

```
          i386: mkbootdisk
```

which means if architecture is i386 then install package *mkbootdisk*

A <required−component1> enforces the dependency from another component and is defined as:

```
@ <component>
```

So, for example, if inside a component definition you find the following line:

```
@ Networked Workstation
```

it means that the component itself needs the installation of another component named *Networked Workstation*.

A <component−dependent−RPM> is used to select the installation of some additional packages for a component, given the presence of another component. Its definition is as follows:

```
? <component> {
  <RPM 1>
  ...
  <RPM n>
}
```

So if, for example, in a component definition, you happen to read the following lines:

```
? KDE {
  kpppload
}
```

then if the *KDE* component is installed, the package *kpppload* will be installed together with the packages included in the component the definition was found in.

7.3. Format of comps file in RedHat version 6.2

With RedHat version 6.2, the format of the `comps` file has, apparently, changed just slightly. The decoding takes place in `${RHROOT}/${ARCH}/misc/src/anaconda/comps.py` even in this case. Once again, I didn't analyze yet this python script and the following rules were obtained only by reading the file and testing some configurations for it.

In release 6.2, the definition of component is extended to include two more optional elements which are:

```
<lang-dependent-RPM 1>
...
<lang-dependent-RPM n>
<arch-dependent-component 1>
...
<arch-dependent-component n>
```

A *<lang−dependent−RPM>* is needed to specify the installation of a package in case a specific language was selected. It's defined as:

```
(lang <language>): <RPM>
```

For example, the following line:

```
        (lang ja_JP)): locale-ja
```

means: if the Japanese language is selected, then install the *locale−ja* package together with the other packages installed for the current component.

An *<arch−dependent−component>* extends the concept of *<arch−dependent−RPM>* introduced in release 6.1 to an entire component, as you can understand reading the definition:

```
        (!)?arch: <component>
```

7.4. Format of comps file in RedHat version 7.3

With RedHat version 7.3, the format of the `comps` file has gained some more syntactical power. The decoding takes place (again) in the `comps.py` script, which you can now find in the `/usr/lib/anaconda/` directory if you have installed the *anaconda* package. The dependencies on a language or an architecture by a component or a package can now be joined with the *and* operator. For example:

```
        (arch !s390 and arch !s390x and arch !ia64): readline2.2.1
```

which means if architecture is not any of s390, s390x, ia64 then install the package readline2.2.1. This can be done with components instead of packages and languages instead of architectures. All this, is definitely more than enough for the simple examples of customization of the default installation which will be presented in the next section.

7.4.1. Customizing the default installation of RedHat version 7.3

The example we will go through in this section implies modifications to the *comps* file to change the default values for packages installation. I usually prefer, in fact, particularly in certain situations a default installation including only the base packages, with some slight alterations to some of them. In the first of the presented examples, we will build a default installation which has the *libsafe* added to the "Base" component and most of the packages which are usually installed by default are deselected, so to build a minimal installation. In the second of the examples, we will modify some of the components to build another minimal installation which fits (this time, almost perfectly) our needs (they are, actually, my needs, your mileage may definitely vary). If you want to include a modified *comps* file in your CDs, you should copy it into the main tree just before starting the operations described in Rebuilding the 7.3/8.0 installer.

7.4.1.1. Adding RPMS and deselecting default components

To customize your installation this way, you have to edit the `comps` file with your favourite text editor (pay attention not to leave harmful spaces or tabs in the file) and move it to the `Redhat/base` directory overwriting the original one.

In the first comps file included, the *libsafe* package was added to the "Base system" component and almost every component was deselected so to have a default installation comprising only two hundred packages (I know they can still be too many...).

7.4.1.2. Modify some of the standard components

In the second comps file included, we build on the previous setup and strip down the default installation a bit more (this time there will be only 154 packages in the default installation). Some of the groups have been splitted to give the installation some more granularity. All the modifications you do should take into account the interdependencies among packages and the applications used during the installation phases (you cannot remove *kudzu*, for example, from the *Base* component, even if you can do it after installation). It must be said that similar results can be obtained using *kickstart*. For more informations about it, you can read The RedHat Linux Customization Guide.

7.5. Format of comps file in RedHat version 8.0 and 9

With RedHat version 8.0 and 9, the format of the `comps` file has changed completely and now an XML file, whose name is `comps.xml`, is used. Details on the file syntax can be found in the anaconda comps section of the RedHat website.

7.5.1. Customizing the default installation of RedHat version 8.0

We will now reproduce the examples presented for release 7.3 taking into account the modifications the various groups were submitted to. The most important group (the "Base" group) is splitted here in two groups which are named "Base" and "Core". The "Base" group should represent the minimal possible installation.

7.5.1.1. Our first example revisited for Redhat 8.0

This time, to customize your installation you have to edit the `comps-milan.xml.in` file with your favourite text editor. This file can be found in the comps−8.0.tar.gz archive found on the Redhat website. To add the packages information to the file you create, you need to have the `comps-extras` rpm package installed. The commands to be issued to complete the operation are listed in Updating comps.xml and in the documentation. After you create the file, you have to copy it to the `Redhat/base` directory overwriting the original one. If you are using the updateBuild.sh script, you should only copy the `comps-milan.xml`, (after having modified the `comps-milan.xml.in` found in the `comps-8.0.tar.gz` tar/gzip package and issued the *make* command), to the destination you should have already configured in the *COMPSFILE* variable (rhcd.conf).

In the first comps file included the *libsafe* package was added to the "Base" group (component) and almost every group (component) was deselected, apart from "Base" and "Core", so to have a default installation comprising only ~220 packages (probably too many, again...).

7.5.1.2. Our second example revisited for Redhat 8.0

In the second comps file included, we build on the previous setup and strip down the default installation a bit more (this time, there will be only 158 packages in the default installation). Once again, similar results can be obtained using *kickstart*, for more informations about it you can read The RedHat Linux Customization Guide. In the example, I didn't unselect completely the installation of the "Base" group, because there are too many packages I usually need, so I just unselected the default installation for these packages making them optional. As you can see, even the `redhat-logos` package in the "Core" group was made optional. Considering that all of the packages in this group, together, should represent the *smallest possible* installation, you probably don't want to do this (by the way my CDs work even with this, there should be some failure I cannot see, yet). The `tripwire` package was also added to the "Base" group. The last noticeable modification was made to the "dialup" group, which will be installed even if unselected because the "Base" group depends on it (as declared in the group definition itself). I have selected only some packages I usually need from this group for installation and left the rest of them unselected.

7.5.2. Customizing the default installation of RedHat version 9

We will reproduce (again) the examples presented for release 7.3/8 taking into account the modifications the various groups were submitted to.

7.5.2.1. Our first example revisited for Redhat 9

As in the case of 8.0, to customize your installation you have to edit the `comps-milan.xml.in` file with your favourite text editor. This file can be found in the comps–9.tar.gz file among the script *(as I said it is not the same you can find on the Redhat website)*. To add the packages information to the file you create, you need to have the `comps-extras` rpm package installed. The commands to be issued to complete the operation are listed in Updating comps.xml and in the documentation. After you create the file, you have to copy it to the `Redhat/base` directory overwriting the original one. If you are using the updateBuild.sh script, you should only copy the `comps-milan.xml`, (after having modified the `comps-milan.xml.in` found in the `comps-9.tar.gz` tar/gzip package and issued the *make* command), to the destination you should have already configured in the *COMPSFILE* variable (rhcd.conf).

In the first comps file included the *libsafe* package was added to the "Base" group (component) and almost every group (component) was deselected, apart from "Base" and "Core", so to have a default installation comprising only ~240 packages (mmmhhh complexity is raising high...).

7.5.2.2. Our second example revisited for Redhat 9

In the second comps file included, we build on the previous setup and strip down the default installation a bit more (this time, there will be only ~175 packages in the default installation). This is really similar to the example presented for Redhat 8.0, so I will avoid boring you with the same explanations. Once again, similar results can be obtained using *kickstart*, for more informations about it you can read The RedHat Linux Customization Guide.

8. Installing from the CD

When installing from the new CD, you may first need to create a bootable installation diskette. *IMPORTANT: use a NEW, freshly MS−DOS formatted diskette!*. Using an old, worn−out, faulty diskette can result in strange problems during the installation! On a Linux system, you can create the diskette using the `dd` command:

```
$ dd if=/mnt/cdrom/images/boot.img of=/dev/fd0 bs=1440k
```

On a system running DOS or Windows−9x, you need to use the `rawrite.exe` program, which is found on the CD in the `dosutils` directory. On a machine with Windows−9x/Me/NT/2k, you can use the `rawritewin.exe` located in the `dosutils/rawritewin` directory.

Shut down the machine you want to install on (or do a system upgrade), insert the boot diskette and your freshly burned CD, and let the machine boot from the diskette. For more information on the installation process, see the documents and the Installation−HOWTO or the Bootdisk−HOWTO.

8.1. Booting from a bootable CD

Most modern machines are able to boot directly from a CD, provided it is made bootable with the procedure outlined in section Creating the CD iso image. Often, however, you need to change the setting of the BIOS to make the CD drive bootable. See the documentation for your mother board to see how it's done.

9. Other Linux distributions

The informations present in the previous versions of the howto (<=1.34), and reported in the current document, which apply to releases up to and including 6.1 of Redhat Linux, is believed to apply to distributions that are Redhat clones such as Mandrake. The procedure is reported to be untested (as you can read in the howto itself) though.

Similar considerations apply to the LinuxPPC distribution for Apple PowerMacs. When making a distribution for the PowerMac platform, you need to use mkhybrid) instead of `mkisofs` and this should be the only difference.

The informations supplied for the new releases of Redhat (>6.1) shouldn't work with Mandrake, which has now a fairly different installer from the Redhat one. I really don't know if some other clone of Redhat can have its distribution CDs updated this way, but I would be happy if you let me know.

10. This document...

The SGML source of the most recent version of this document can be retrieved from here. The previous version, created by Morten Kjeldgaard, and Peter von der Ahé, can be found on imsb.au.dk

10.1. Related documentation

10.1.1. Documentation related to the current version

The following documents were useful in the creation of this howto:

The (unofficial) RedHat 7 Customised Installer mini−HOWTO, by Tony Nugent. This document is very interesting and useful, so, if you are serious about building customized CDs, I strongly suggest you to read it. You can find it on www.linuxworks.com.au

Miguel Freitas has written RedHat7 CDs mini−Howto, that you can read on this website.

Ron Yorston wrote the rpmhack document, relevant for release 6.2 of Redhat Linux.

Someone (I couldn't find his name) wrote the document Building a Red Hat Linux 6.2 CDROM, useful for release 6.2.

10.1.2. Documentation related to the previous edition

Ed Schlunder <zilym@asu.edu> has written a utility called `fix−rhcd` to let you check your Red Hat Linux distribution mirror for matching file sizes, names, permissions, and symlinks against an "ls −lNR" listing from the offical Red Hat ftp site. Any permissions that are wrong are changed to match the "ls" listing. See the fix−rhcd homepage.

Rod Smith <smithrod@bellatlantic.net> has written a Do−It−Yourself Red Hat Installation guide, which also includes information on creating RedHat install CD's. Especially aimed at burning a CD from a non−UNIX system. Find it on his website.

A document in french "Comment graver un CD de la RedHat 5.x a partir de fichiers telecharges sur Internet..."" by <skooter@hol.fr> is available from linuxfr.org.

With the sense of the good things in life Jussi Torhonen from Finland <jussi.torhonen@tietosavo.fi> tells us Howto make a homebrew bootable RedHat Linux 5.2 CD−ROM.

From the LDP project, see the CD−writing HOWTO.

10.2. Acknowledgements

Apart from those mentioned above, thanks are given to the following people for valuable input, feedback, discussions and other:

10.2.1. Acknowledgements for the current version

- Morten Kjeldgaard, <mok (at) imsb (dot) au (dot) dk>
- Peter von der Ahé, <pahe+rhcd (at) daimi (dot) au (dot) dk>
- Giulia Tomaselli
- Jacinta Conneely
- Filippo Carcaci
- Guillaume Lelarge <gleu (at) wanadoo (dot) fr>
- Alain Portal <aportal (at) univ−montp2 (dot) fr>
- All the people on the anaconda−devel and kickstart mailing lists

---

10.2.2. Acknowledgements for the previous versions

- Lars Christensen <larsch (at) cs (dot) auc (dot) dk>
- Thomas Duffy <tbd (at) cs (dot) brown (dot) edu>
- Dawn Endico <dawn (at) math (dot) wayne (dot) edu>
- Seva <seva (at) null (dot) cc (dot) uic (dot) edu>
- Michael Thomas Cope <mcope (at) orion (dot) ac (dot) hmc (dot) edu>
- Charles J. Fisher <charles_fisher (at) bigfoot (dot) com>
- Eric Thomas <eric.thomas (at) ericsson (dot) com>
- Gordon Yuen <gdccyuen (at) yahoo (dot) com>
- Dave Morse <morse (at) nichimen (dot) com>