NAME

reprepro − produce, manage and sync a local repository of Debian packages

SYNOPSIS

reprepro −−help

reprepro [*options*] *command* [*per−command−arguments*]

DESCRIPTION

reprepro is a tool to manage a repository of Debian packages (.deb, .udeb, .dsc, ...). It stores files either being injected manually or downloaded from some other repository (partially) mirrored into a pool/ hierarchy. Managed packages and checksums of files are stored in a Berkeley DB database file, so no database server is needed. Checking signatures of mirrored repositories and creating signatures of the generated Package indices is supported.

Former working title of this program was mirrorer.

GLOBAL OPTIONS

Options can be specified before the command. Each affects a different subset of commands and is ignored by other commands.

−h −−help
Displays a short list of options and commands with description.

−v, −V, −−verbose
Be more verbose. Can be applied multiple times. One uppercase **−V** counts as five lowercase **−v.**

−−silent
Be less verbose. Can be applied multiple times. One **−v** and one **−s** cancel each other out.

−f, −−force
This option is ignored, as it no longer exists.

−b, −−basedir *basedir*
Sets the base−dir all other default directories are relative to. If none is supplied and the **REPREPRO_BASE_DIR** environment variable is not set either, the current directory will be used.

−−outdir *outdir*
Sets the base−dir of the repository to manage, i.e. where the **pool/** subdirectory resides. And in which the **dists/** directory is placed by default. If this starts with '**+b/**', it is relative to basedir.

The default for this is *basedir*.

−−confdir *confdir*
Sets the directory where the configuration is searched in.

If this starts with '**+b/**', it is relative to basedir.

If none is given, **+b/conf** (i.e. *basedir*/**conf**) will be used.

−−distdir *distdir*
Sets the directory to generate index files relatively to. (i.e. things like Packages.gz, Sources.gz and Release.gpg)

If this starts with '**+b/**', it is relative to basedir, if starting with '**+o/**' relative to outdir.

If none is given, **+o/dists** (i.e. *outdir*/**dists**) is used.

Note: apt has **dists** hard-coded in it, so this is mostly only useful for testing or when your webserver pretends another directory structure than your physical layout.

Warning: Beware when changing this forth and back between two values not ending in the same directory. Reprepro only looks if files it wants are there. If nothing of the content changed and there is a file it will not touch it, assuming it is the one it wrote last time, assuming any different **−−distdir** ended in the same directory. So either clean a directory before setting**−−distdir** to it or do an **export** with the new one first to have a consistent state.

−−logdir *logdir*
The directory where files generated by the **Log:** directive are stored if they have no absolute path.

If this starts with '**+b/**', it is relative to basedir, if starting with '**+o/**' relative to outdir, with '**+c/**' relative to confdir.

If none is given, **+b/logs** (i.e. *basedir*/**logs**) is used.

−−dbdir *dbdir*
Sets the directory where reprepro keeps its databases.

If this starts with '**+b/**', it is relative to basedir, if starting with '**+o/**' relative to outdir, with '**+c/**' relative to confdir.

If none is given, **+b/db** (i.e. *basedir*/**db**) is used.

Note: This is permanent data, no cache. One has almost to regenerate the whole repository when this is lost.

−−listdir *listdir*
Sets the directory where it downloads indices to when importing from other repositories. This is temporary data and can be safely deleted when not in an update run.

If this starts with '**+b/**', it is relative to basedir, if starting with '**+o/**' relative to outdir, with '**+c/**' relative to confdir.

If none is given, **+b/lists** (i.e. *basedir*/**lists**) is used.

−−morguedir *morguedir*
Files deleted from the pool are stored into *morguedir*.

If this starts with '**+b/**', it is relative to basedir, if starting with '**+o/**' relative to outdir, with '**+c/**' relative to confdir.

If none is given, deleted files are just deleted.

−−methoddir *methoddir*
Look in *methoddir* instead of **/usr/lib/apt/methods** for methods to call when importing from other repositories.

−C, −−component *components*
Limit the specified command to this components only. This will force added packages to this components, limit removing packages from this components, only list packages in this components, and/or otherwise only look at packages in this components, depending on the command in question.

Multiple components are specified by separating them with |, as in **−C 'main|contrib'**.

−A, −−architecture *architectures*
Limit the specified command to this architectures only. (i.e. only list such packages, only remove packages from the specified architectures, or otherwise only look at/act on this architectures depending on the specific command).

Multiple architectures are specified by separating them with |, as in **−A 'sparc|i386'**.

Note that architecture **all** packages can be included to each architecture but are then handled separately. Thus by using **−A** in a specific w ay one can have different versions of an architecture **all** package in different architectures of the same distribution.

−T, −−type dsc|deb|udeb
Limit the specified command to this packagetypes only. (i.e. only list such packages, only remove such packages, only include such packages, ...)

−S, −−section *section*
Overrides the section of inclusions. (Also override possible override files)

−P, −−priority *priority*
Overrides the priority of inclusions. (Also override possible override files)

−−export=(silent−never|never|changed|lookedat|force)
This option specify whether and how the high level actions (e.g. install, update, pull, delete) should export the index files of the distributions they work with.

−−export=lookedat
In this mode every distribution the action handled will be exported, unless there was an error possibly corrupting it.
Note that only missing files and files whose intended content changed between before and after the action will be written. To get a guaranteed current export, use the **export** action.
For backwards compatibility, **lookedat** is also available under the old name **normal**. The name **normal** is deprecated and will be removed in future versions.

−−export=changed
In this mode every distribution actually changed will be exported, unless there was an error possibly corrupting it. (i.e. if nothing changed, not even missing files will be created.)
Note that only missing files and files whose intended content changed between before and after the action will be written. To get a guaranteed current export, use the **export** action.

−−export=force
Always export all distributions looked at, even if there was some error possibly bringing it into a inconsistent state.

−−export=never
No index files are exported. You will have to call **export** later.
Note that you most likely additionally need the **−−keepunreferencedfiles** option, if you do not want some of the files pointed to by the untouched index files to vanish.

−−export=silent-never
Like never, but suppress most output about that.

−−ignore=*what*
Ignore errors of type *what*. See the section **ERROR IGNORING** for possible values.

−−nolistsdownload
When running **update**, **checkupdate** or **predelete** do not download any Release or index files. This is hardly useful except when you just run one of those command for the same distributions. And even then reprepro is usually good in not downloading except **Release** and **Release.gpg** files again.

−−nothingiserror
If nothing was done, return with exitcode 1 instead of the usual 0.

Note that "nothing was done" means the primary purpose of the action in question. Auxiliary actions (opening and closing the database, exporting missing files with −−export=lookedat, ...) usually do not count. Also note that this is not very well tested. If you find an action that claims to have done something in some cases where you think it should not, please let me know.

−−**keeptemporaries**
Do not delete temporary **.new** files when exporting a distribution fails. (reprepro first create**.new** files in the **dists** directory and only if everything is generated, all files are put into their final place at once. If this option is not specified and something fails, all are deleted to keep **dists** clean).

−−**keepunreferencedfiles**
Do not delete files that are no longer used because the package they are from is deleted/replaced with a newer version from the last distribution it was in.

−−**keepunusednewfiles**
The include, includedsc, includedeb and processincoming by default delete any file they added to the pool that is not marked used at the end of the operation. While this keeps the pool clean and allows changing before trying to add again, this needs copying and checksum calculation every time one tries to add a file.

−−**keepdirectories**
Do not try to rmdir parent directories after files or directories have been removed from them. (Do this if your directories have special permissions you want keep, do not want to be pestered with warnings about errors to remove them, or have a buggy rmdir call deleting non-empty directories.)

−−**ask−passphrase**
Ask for passphrases when signing things and one is needed. This is a quick and dirty and unsafe implementation using the obsolete **getpass(3)** function with the description gpgme is supplying. So the prompt will look quite funny and support for passphrases with more than 8 characters depend on your libc. Use of this option is not recommended. Use gpg-agent with pinentry instead.

(With current versions of gnupg you need to set **pinentry-mode loopback** in your .gnupg/gpg.conf file to use −−**ask−passphrase**. Without that option gnupg uses the much safer and recommended pinentry instead).

−−**noskipold**
When updating do not skip targets where no new index files and no files marked as already processed are available.

If you changed a script to preprocess downloaded index files or changed a Listfilter, you most likely want to call reprepro with −−noskipold.

−−**waitforlock** *count*
If there is a lockfile indicating another instance of reprepro is currently using the database, retry *count* times after waiting for 10 seconds each time. The default is 0 and means to error out instantly.

−−**spacecheck full|none**
The default is **full**:
In the update commands, check for every to be downloaded file which filesystem it is on and how much space is left.
To disable this behaviour, use **none**.

−−**dbsafetymargin** *bytes-count*
If checking for free space, reserve *byte-count* bytes on the filesystem containing the **db/** directory. The default is 104857600 (i.e. 100MB), which is quite large. But as there is no way to know in advance how large the databases will grow and libdb is extremely touchy in that regard, lower only when you know what you do.

−−**safetymargin** *bytes-count*
If checking for free space, reserve *byte-count* bytes on filesystems not containing the **db/** directory. The default is 1048576 (i.e. 1MB).

−−noguessgpgtty
Don't set the environment variable **GPG_TTY**, even when it is not set, stdin is terminal and **/proc/self/fd/0** is a readable symbolic link.

−−gnupghome
Set the **GNUPGHOME** evnironment variable to the given directory as argument to this option. And your gpg will most likely use the content of this variable instead of "~/.gnupg". Take a look at **gpg**(1) to be sure. This option in the command line is usually not very useful, as it is possible to set the environment variable directly. Its main reason for existence is that it can be used in *conf*/**options**.

−−gunzip *gz-uncompressor*
While reprepro links against **libz**, it will look for the program given with this option (or **gunzip** if not given) and use that when uncompressing index files while downloading from remote repositories. (So that downloading and uncompression can happen at the same time). If the program is not found or is **NONE** (all-uppercase) then uncompressing will always be done using the built in uncompression method. The program has to accept the compressed file as stdin and write the uncompressed file into stdout.

−−bunzip2 *bz2-uncompressor*
When uncompressing downloaded index files or when not linked against **libbz2** reprepro will use this program to uncompress **.bz2** files. The default value is **bunzip2**. If the program is not found or is **NONE** (all-uppercase) then uncompressing will always be done using the built in uncompression method or not be possible when not linked against **libbz2**. The program has to accept the compressed file as stdin and write the uncompressed file into stdout.

−−unlzma *lzma-uncompressor*
When trying to uncompress or read lzma compressed files, this program will be used. The default value is **unlzma**. If the program is not found or is**NONE** (all-uppercase) then uncompressing lzma files will not be possible. The program has to accept the compressed file as stdin and write the uncompressed file into stdout.

−−unxz *xz-uncompressor*
When trying to uncompress or read xz compressed files, this program will be used. The default value is **unxz**. If the program is not found or is**NONE** (all-uppercase) then uncompressing xz files will not be possible. The program has to accept the compressed file as stdin and write the uncompressed file into stdout.

−−lunzip *lzip-uncompressor*
When trying to uncompress or read lzip compressed files, this program will be used. The default value is **lunzip**. If the program is not found or is**NONE** (all-uppercase) then uncompressing lz files will not be possible. The program has to accept the compressed file as stdin and write the uncompressed file into stdout. Note that .lz support is **DEPRECATED** and will be removed in the future.

−−list−max *count*
Limits the output of **list**, **listmatched** and **listfilter** to the first *count* results. The default is 0, which means unlimited.

−−list−skip *count*
Omitts the first *count* results from the output of **list**, **listmatched** and **listfilter**.

−−list−format *format*
Set the output format of **list**, **listmatched** and **listfilter** commands. The format is similar to dpkg−query's **−−showformat**: fields are specified as **${** *fieldname***}** or **${** *fieldname***;**_length_**}**. Zero length or no length means unlimited. Positive numbers mean fill with spaces right, negative fill with spaces left.

\n, **\r**, **\t**, **\0** are new-line, carriage-return, tabulator and zero-byte. Backslash (\) can be used to escape every non-letter-or-digit.

The special field names **$identifier**, **$architecture**, **$component**, **$type**, **$codename** denote where the package was found.

The special field names **$source** and **$sourceversion** denote the source and source version a package belongs to. (i.e. **${$sour ce}** will either be the same as **${source}** (without a possible version in parentheses at the end) or the same as **${package}**.

The special field names **$basename**, **$filekey** and **$fullfilename** denote the first package file part of this entry (i.e. usually the .deb, .udeb or .dsc file) as basename, as filekey (filename relative to the outdir) and the full filename with outdir prepended (i.e. as relative or absolute as your outdir (or basedir if you did not set outdir) is).

When **−−list−format** is not given or **NONE**, then the default is equivalent to
${$identifier} ${package} ${version}\n.

Escaping digits or letters not in above list, using dollars not escaped outside specified constructs, or any field names not listed as special and not consisting entirely out of letters, digits and minus signs have undefined behaviour and might change meaning without any further notice.

If you give this option on the command line, don't forget that $ is also interpreted by your shell. So you have to properly escape it. For example by putting the whole argument to −−list−format in single quotes.

−−show−percent
When downloading packages, show each completed percent of completed package downloads together with the size of completely downloaded packages. (Repeating this option increases the frequency of this output).

−−onlysmalldeletes
The pull and update commands will skip every distribution in which one target loses more than 20% of its packages (and at least 10).

Using this option (or putting it in the options config file) can avoid removing large quantities of data but means you might often give **−−noonlysmalldeletes** to override it.

−−restrict *src*[**=***version*|**:***type*]
Restrict a **pull** or **update** to only act on packages belonging to source-package *src*. Any other package will not be updated (unless it matches a **−−restrict−bin**). Only packages that would otherwise be updated or are at least marked with **hold** in a **FilterList** or **FilerSrcList** will be updated.

The action can be restricted to a source version using a equal sign or changed to another type (see FilterList) using a colon.

This option can be given multiple times to list multiple packages, but each package may only be named once (even when there are different versions or types).

−−restrict−binary *name*[**=***version*|**:***type*]
Like **−−restrict** but restrict to binary packages (**.deb** and **.udeb**). Source packages are not upgraded unless they appear in a **−−restrict**.

−−restrict−file *filename*
Like **−−restrict** but read a whole file in the **FilterSrcList** format.

−−restrict−file−bin *filename*
Like **−−restrict−bin** but read a whole file in the **FilterList** format.

−−endhook *hookscript*

Run the specified *hookscript* once reprepro exits. It will get the usual**REPREPR O_*

environment variables set (or unset) and additionally a variable **REPREPRO_EXIT_CODE** that is the exit code with which reprepro would have exited (the hook is always called once the initial parsing of global options and the command name is done, no matter if reprepro did anything or not). Reprepro will return to the calling process with the exitcode of this script. Reprepro has closed all its databases and removed all its locks, so you can run reprepro again in this script (unless someone else did so in the same repository before, of course).

The only advantage over running that command always directly after reprepro is that you can some environment variables set and cannot so easily forget it if this option is in conf/options.

The script is supposed to be located relative to *confdir*, unless its name starts with **/**, **./**, **+b/**, **+o/**, or **+c/** and the name may not start (except in the cases given before) with a **+**.

An example script looks like:

```
#!/bin/sh

if [ "$REPREPRO_EXIT_CODE" -ne 0 ] ; then
        exit "$REPREPRO_EXIT_CODE"
fi

echo "congratulations, reprepro with arguments: $*"
echo "seems to have run successfully. REPREPRO_ part of the environment is:"
set | grep ^REPREPRO_

exit 0
```

−−outhook *hookscript*

hookscript is called with a **.outlog** file as argument (located in *logdir*) containing a description of all changes made to *outdir*.

The script is supposed to be located relative to *confdir*, unless its name starts with **/**, **./**, **+b/**, **+o/**, or **+c/** and the name may not start (except in the cases given before) with a **+**.

For a format of the **.outlog** files generated for this script see the **manual.html** shiped with reprepro.

COMMANDS

export [*codenames*]

Generate all index files for the specified distributions.

This regenerates all files unconditionally. It is only useful if you want to be sure **dists** is up to date, you called some other actions with **−−export=never** before or you want to create an initial empty but fully equipped **dists/***codename* directory.

[**−−delete**] **createsymlinks** [*codenames*]

Creates *suite* symbolic links in the **dists/**-directory pointing to the corresponding *codename*.

It will not create links, when multiple of the given codenames would be linked from the same suite name, or if the link already exists (though when **−−delete** is given it will delete already existing symlinks)

list *codename* [*packagename*]

List all packages (source and binary, except when **−T** or **−A** is given) with the given name in all components (except when **−C** is given) and architectures (except when **−A** is given) of the specified distribution. If no package name is given, list everything. The format of the output can be changed with **−−list−format**. To only get parts of the result, use **−−list−max** and **−−list−skip**.

listmatched *codename glob*
as list, but does not list a single package, but all packages matching the given shell-like *glob*. (i.e. *****, **?** and **[***chars***]** are allowed).

Examples:

reprepro −b . listmatched test2 'linux−*' lists all packages starting with **linux−**.

listfilter *codename condition*
as list, but does not list a single package, but all packages matching the given condition.

The format of the formulas is those of the dependency lines in Debian packages' control files with some extras. That means a formula consists of names of fields with a possible condition for its content in parentheses. These atoms can be combined with an exclamation mark '!' (meaning not), a pipe symbol '|' (meaning or) and a comma ',' (meaning and). Additionally parentheses can be used to change binding (otherwise '!' binds more than '|' than ',').

The values given in the search expression are directly alphabetically compared to the headers in the respective index file. That means that each part *Fieldname* **(***cmp value***)** of the formula will be true for exactly those package that have in the **Package** or **Sources** file a line starting with *fieldname* and a value is alphabetically *cmp* to *value*.

Additionally since reprepro 3.11.0, '**%**' can be used as comparison operator, denoting matching a name with shell like wildcard (with '*****', '**?**' and '**[..]**').

The special field names starting with '**$**' have special meaning (available since 3.11.1):

$Version

The version of the package, comparison is not alphabetically, but as Debian version strings.

$Source

The source name of the package.

$SourceVersion

The source version of the package.

$Architecture

The architecture the package is in (listfilter) or to be put into.

$Component

The component the package is in (listfilter) or to be put into.

$Packagetype

The packagetype of the package.

Examples:

reprepro −b . listfilter test2 'Section (== admin)' will list all packages in distribution test2 with

a Section field and the value of that field being **admin**.

reprepro −b . −T deb listfilter test2 'Source (== *blub*) | **(!Source , Package (**== *blub*))' will find all .deb Packages with either a Source field blub or no Source field and a Package field blub. (That means all package generated by a source package *blub*, except those also specifying a version number with its Source).

reprepro −b . −T deb listfilter test2 '$Source (==*blub*) is the better way to do this (but only available since 3.11.1).

reprepro −b . listfilter test2 '$PackageType (==deb), $Source (==*blub*) is another (less efficient) way.

reprepro −b . listfilter test2 'Package (% linux−*−2.6*)' lists all packages with names starting with **linux−** and later having an **−2.6**.

ls *package-name*
List the versions of the specified package in all distributions.

lsbycomponent *package-name*
Like ls, but group by component (and print component names).

remove *codename package-names*
Delete all packages in the specified distribution, that have package name listed as argument. (i.e. remove all packages **list** with the same arguments and options would list, except that an empty package list is not allowed.)

Note that like any other operation removing or replacing a package, the old package's files are unreferenced and thus may be automatically deleted if this was their last reference and no **−−keepunreferencedfiles** specified.

removematched *codename glob*
Delete all packages **listmatched** with the same arguments would list.

removefilter *codename condition*
Delete all packages **listfilter** with the same arguments would list.

removesrc *codename source-name* [*version*]
Remove all packages in distribution *codename* belonging to source package *source-name*. (Limited to those with source version *version* if specified).

If package tracking is activated, it will use that information to find the packages, otherwise it traverses all package indices for the distribution.

removesrcs *codename source-name*[=*version*] ...
Like **removesrc**, but can be given multiple source names and source versions must be specified by appending '=' and the version to the name (without spaces).

update [*codenames*]
Sync the specified distributions (all if none given) as specified in the config with their upstreams. See the description of **conf/updates** below.

checkupdate [*codenames*]
Same like **update**, but will show what it will change instead of actually changing it.

dumpupdate [*codenames*]
Same like **checkupdate**, but less suiteable for humans and more suitable for computers.

predelete [*codenames*]
This will determine which packages a **update** would delete or replace and remove those packages. This can be useful for reducing space needed while upgrading, but there will be some time where

packages are vanished from the lists so clients will mark them as obsolete. Plus if you cannot download a updated package in the (hopefully) following update run, you will end up with no package at all instead of an old one. This will also blow up **.diff** files if you are using the pdiff example or something similar. So be careful when using this option or better get some more space so that update works.

cleanlists
Delete all files in *listdir* (default *basedir*/**lists**) that do not belong to any update rule for any distribution. I.e. all files are deleted in that directory that no**update** command in the current configuration can use. (The files are usually left there, so if they are needed again they do not need to be downloaded again. Though in many easy cases not even those files will be needed.)

pull [*codenames*]
pull in newer packages into the specified distributions (all if none given) from other distributions in the same repository. See the description of**conf/pulls** belo w.

checkpull [*codenames*]
Same like **pull**, but will show what it will change instead of actually changing it.

dumppull [*codenames*]
Same like **checkpull**, but less suiteable for humans and more suitable for computers.

includedeb *codename .deb-filename*
Include the given binary Debian package (.deb) in the specified distribution, applying override information and guessing all values not given and guessable.

includeudeb *codename .udeb-filename*
Same like **includedeb**, but for .udeb files.

includedsc *codename .dsc-filename*
Include the given Debian source package (.dsc, including other files like .orig.tar.gz, .tar.gz and/or .diff.gz) in the specified distribution, applying override information and guessing all values not given and guessable.

Note that .dsc files do not contain section or priority, but the Sources.gz file needs them. reprepro tries to parse .diff and .tar files for it, but is only able to resolve easy cases. If reprepro fails to extract those automatically, you have to either specify a DscOverride or give them via **−S** and **−P**

include *codename .changes-filename*
Include in the specified distribution all packages found and suitable in the *.changes* file, applying override information guessing all values not given and guessable.

processincoming *rulesetname* [*.changes-file*]
Scan an incoming directory and process the .changes files found there. If a filename is supplied, processing is limited to that file. *rulesetname* identifies which rule-set in **conf/incoming** determines which incoming directory to use and in what distributions to allow packages into. See the section about this file for more information.

check [*codenames*]
Check if all packages in the specified distributions have all files needed properly registered.

checkpool [**fast**]
Check if all files believed to be in the pool are actually still there and have the known md5sum. When **fast** is specified md5sum is not checked.

collectnewchecksums
Calculate all supported checksums for all files in the pool. (Versions prior to 3.3 did only store md5sums, 3.3 added sha1, 3.5 added sha256).

translatelegacychecksums
Remove the legacy **files.db** file after making sure all information is also found in the new **checksums.db** file. (Alternatively you can call **collecnewchecksums** and remove the file on your own.)

rereference
Forget which files are needed and recollect this information.

dumpreferences
Print out which files are marked to be needed by whom.

dumpunreferenced
Print a list of all filed believed to be in the pool, that are not known to be needed.

deleteunreferenced
Remove all known files (and forget them) in the pool not marked to be needed by anything.

deleteifunreferenced [*filekeys*]
Remove the given files (and forget them) in the pool if they are not marked to be used by anything. If no command line arguments are given, stdin is read and every line treated as one filekey. This is mostly useful together with **−−keepunreferenced** in **conf/options** or in situations where one does not want to run **deleteunreferenced**, which removes all files eligible to be deleted with this command.

reoverride [*codenames*]
Reapply the override files to the given distributions (Or only parts thereof given by **−A**,**−C** or **−T**).

Note: only the control information is changed. Changing a section to a value, that would cause another component to be guessed, will not cause any warning.

redochecksums [*codenames*]
Readd the information about file checksums to the package indices.

Usually the package's control information is created at inclusion time or imported from some remote source and not changed later. This command modifies it to readd missing checksum types.

Only checksums already known are used. To update known checkums about files run **collectnewchecksums** first.

dumptracks [*codenames*]
Print out all information about tracked source packages in the given distributions.

retrack [*codenames*]
Recreate a tracking database for the specified distributions. This contains ouf of three steps. First all files marked as part of a source package are set to unused. Then all files actually used are marked as thus. Finally tidytracks is called remove everything no longer needed with the new information about used files.

(This behaviour, though a bit longsome, keeps even files only kept because of tracking mode **keep** and files not otherwise used but kept due to **includechanges** or its relatives. Before version 3.0.0 such files were lost by running retrack).

removealltracks [*codenames*]
Removes all source package tracking information for the given distributions.

removetrack *codename sourcename version*
Remove the trackingdata of the given version of a given sourcepackage from a given distribution. This also removes the references for all used files.

tidytracks [*codenames*]
Check all source package tracking information for the given distributions for files no longer to keep.

copy *destination-codename source-codename packages...*
Copy the given packages from one distribution to another. The packages are copied verbatim, no override files are consulted. Only components and architectures present in the source distribution

are copied.

copysrc *destination-codename source-codename source-package* [*versions*]
look at each package (where package means, as usual, every package be it dsc, deb or udeb) in the distribution specified by *source-codename* and identifies the relevant source package for each. All packages matching the specified *source-package* name (and any *version* if specified) are copied to the *destination-codename* distribution. The packages are copied verbatim, no override files are consulted. Only components and architectures present in the source distribution are copied.

copymatched *destination-codename source-codename glob*
Copy packages matching the given glob (see **listmatched**).

The packages are copied verbatim, no override files are consulted. Only components and architectures present in the source distribution are copied.

copyfilter *destination-codename source-codename formula*
Copy packages matching the given formula (see **listfilter**). (all versions if no version is specified). The packages are copied verbatim, no override files are consulted. Only components and architectures present in the source distribution are copied.

restore *codename snapshot packages...*

restoresrc *codename snapshot source-epackage* [*versions*]

restorefilter *destination-codename snapshot formula*

restorematched *destination-codename snapshot glob*
Like the copy commands, but do not copy from another distribution, but from a snapshot generated with **gensnapshot**. Note that this blindly trusts the contents of the files in your**dists/** directory and does no checking.

clearvanished
Remove all package databases that no longer appear in **conf/distributions**. If−−**delete** is specified, it will not stop if there are still packages left. Even without −−**delete** it will unreference files still marked as needed by this target. (Use−−**k eepunreferenced** to not delete them if that was the last reference.)

Do not forget to remove all exported package indices manually.

gensnapshot *codename directoryname*
Generate a snapshot of the distribution specified by *codename* in the directory *dists*/*codename*/**snapshots/***directoryname*/ and reference all needed files in the pool as needed by that. No Content files are generated and no export hooks are run.

Note that there is currently no automated way to remove that snapshot again (not even clearvanished will unlock the referenced files after the distribution itself vanished). You will have to remove the directory yourself and tell reprepro to **unreferencesnapshot** *codename directoryname* before **deleteunreferenced** will delete the files from the pool locked by this.

To access such a snapshot with apt, add something like the following to your sources.list file:
deb method://as/without/snapshot *codename*/**snapshots/***name* **main**

unreferencesnapshot *codename directoryname*
Remove all references generated by an **genshapshot** with the same arguments. This allows the next **deleteunferenced** call to delete those files. (The indicies in **dists/** for the snapshot are not removed.)

rerunnotifiers [*codenames*]
Run all external scripts specified in the **Log:** options of the specified distributions.

build−needing *codename architecture* [*glob*]
List source packages (matching *glob*) that likely need a build on the given architecture.

List all source package in the given distribution without a binary package of the given architecture built from that version of the source, without a **.changes** or **.log** file for the given architecture, with an Architecture field including **any**, *os*-**any** (with *os* being the part before the hyphen in the architecture or **linux** if there is no hyphen) or the architecture and at least one package in the Binary field not yet available.

If instead of *architecture* the term **any** is used, all architectures are iterated and the architecture is printed as fourth field in every line.

If the *architecture* is **all**, then only source packages with an Architecture field including **all** are considered (i.e. as above with real architectures but **any** does not suffice). Note that dpkg−dev << 1.16.1 does not both set **any** and **all** so source packages building both architecture dependent and independent packages will never show up unless built with a new enough dpkg−source).

translatefilelists
Translate the file list cache within *db*/**contents.cache.db** into the new format used since reprepro 3.0.0.

Make sure you have at least half of the space of the current *db*/**contents.cache.db** file size available in that partition.

flood *distribution* [*architecture*]
For each architecture of *distribution* (or for the one specified) add architecture **all** packages from other architectures (but the same component or packagetype) under the following conditions:

Packages are only upgraded, never downgraded.
If there is a package not being architecture **all**, then architecture **all** packages of the same source from the same source version are preferred over those that have no such binary sibling.
Otherwise the package with the highest version wins.

You can restrict with architectures are looked for architecture **all** packages using **−A** and which components/packagetypes are flooded by **−C**/**−T** as usual.

There are mostly two use cases for this command: If you added an new architecture to an distribution and want to copy all architecture **all** packages to it. Or if you included some architecture all packages only to some architectures using **−A** to avoid breaking the other architectures for which the binary packages were still missing and now want to copy it to those architectures were they are unlikely to break something (because a newbinary is already available).

unusedsources [*distributions*]
List all source packages for which no binary package build from them is found.

sourcemissing [*distributions*]
List all binary packages for which no source package is found (the source package must be in the same distribution, but source packages only kept by package tracking is enough).

reportcruft [*distributions*]
List all source package versions that either have a source package and no longer a binary package or binary packages left without source package in the index. (Unless sourcemissing also list packages where the source package in only in the pool due to enabled tracking but no longer in the index).

sizes [*codenames*]
List the size of all packages in the distributions specified or in all distributions.

Each row contains 4 numbers, each being a number of bytes in a set of packages, which are: The packages in this distribution (including anything only kept because of tracking), the packages only in this distribution (anything in this distribution and a snapshot of this distribution counts as only in this distribution), the packages in this distribution and its snapshots, the packages only in this distribution or its snapshots.

If more than one distribution is selected, also list a sum of those (in which 'Only' means only in selected ones, and not only only in one of the selected ones).

repairdescriptions [*codenames*]
Look for binary packages only having a short description and try to get the long description from the .deb file (and also remove a possible Description-md5 in this case).

internal commands
These are hopefully never needed, but allow manual intervention. **WARNING:** Is is quite easy to get into an inconsistent and/or unfixable state.

_detect [*filekeys*]
Look for the files, which *filekey* is given as argument or as a line of the input (when run without arguments), and calculate their md5sum and add them to the list of known files. (Warning: this is a low level operation, no input validation or normalization is done.)

_forget [*filekeys*]
Like **_detect** but remove the given *filekey* from the list of known files. (Warning: this is a low level operation, no input validation or normalization is done.)

_listmd5sums
Print a list of all known files and their md5sums.

_listchecksums
Print a list of all known files and their recorded checksums.

_addmd5sums
alias for the newer

_addchecksums
Add information of known files (without any check done) in the strict format of _listchecksums output (i.e. don't dare to use a single space anywhere more than needed).

_dumpcontents *identifier*
Printout all the stored information of the specified part of the repository. (Or in other words, the content the corresponding Packages or Sources file would get)

_addreference *filekey identifier*
Manually mark *filekey* to be needed by *identifier*

_addreferences *identifier* [*filekeys*]
Manually mark one or more *filekeys* to be needed by *identifier*. If no command line arguments are given, stdin is read and every line treated as one filekey.

_removereference *identifier filekey*
Manually remove the given mark that the file is needed by this identifier.

_removereferences *identifier*
Remove all references what is needed by *identifier.*

__extractcontrol *.deb-filename*
Look what reprepro believes to be the content of the **control** file of the specified .deb-file.

__extractfilelist *.deb-filename*
Look what reprepro believes to be the list of files of the specified .deb-file.

_fakeemptyfilelist *filekey*
Insert an empty filelist for *filekey*. This is a evil hack around broken .deb files that cannot be read by reprepro.

_addpackage *codenam filename packages...*
Add packages from the specified filename to part specified by **−C** **−A** and **−T** of the specified distribution. Very strange things can happen if you use it improperly.

__dumpuncompressors
List what compressions format can be uncompressed and how.

__uncompress *format compressed-file uncompressed-file*
Use builtin or external uncompression to uncompress the specified file of the specified format into the specified target.

_listcodenames
Print - on per line - the codenames of all configured distributions.

_listconfidentifiers *identifier* [*distributions...*]
Print - one per line - all identifiers of subdatabases as derived from the configuration. If a list of distributions is given, only identifiers of those are printed.

_listdbidentifiers *identifier* [*distributions...*]
Print - one per line - all identifiers of subdatabases in the current database. This will be a subset of the ones printed by **_listconfidentifiers** or most commands but **clearvanished** will refuse to run, and depending on the database compatibility version, will include all those if reprepro was run since the config was last changed.

CONFIG FILES

reprepo uses three config files, which are searched in the directory specified with **−−confdir** or in the **conf/** subdirectory of the *basedir*.

If a file **options** exists, it is parsed line by line. Each line can be the long name of a command line option (without the −−) plus an argument, where possible. Those are handled as if they were command line options given before (and thus lower priority than) any other command line option. (and also lower priority than any environment variable).

To allow command line options to override options file options, most boolean options also have a corresponding form starting with **−−no**.

(The only exception is when the path to look for config files changes, the options file will only opened once and of course before any options within the options file are parsed.)

The file **distributions** is always needed and describes what distributions to manage, while **updates** is only needed when syncing with external repositories and **pulls** is only needed when syncing with repositories in the same reprepro database.

The last three are in the format control files in Debian are in, i.e. paragraphs separated by empty lines consisting of fields. Each field consists of a fieldname, followed by a colon, possible whitespace and the data. A field ends with a newline not followed by a space or tab.

Lines starting with # as first character are ignored, while in other lines the # character and everything after it till the newline character are ignored.

A paragraph can also consist of only a single field "**!include:**" which causes the named file (relative to confdir unless starting with **~/**, **+b/**, **+c/** or **/**) to be read as if it was found at this place.

Each of the three files or a file included as described above can also be a directory, in which case all files it contains with a filename ending in **.conf** and not starting with **.** are read.

conf/distributions

Codename

This required field is the unique identifier of a distribution and used as directory name within **dists/** It is also copied into the Release files.

Note that this name is not supposed to change. You most likely **never ever** want a name like **testing** or **stable** here (those are suite names and supposed to point to another distribution later).

Suite This optional field is simply copied into the Release files. In Debian it contains names like stable, testing or unstable. To create symlinks from the Suite to the Codename, use the **createsymlinks** command of reprepro.

FakeComponentPrefix

If this field is present, its argument is added - separated by a slash - before every Component written to the main Release file (unless the component already starts with it), and removed from the end of the Codename and Suite fields in that file. Also if a component starts with it, its directory in the dists dir is shortened by this.
So

Codename: bla/updates
Suite: foo/updates
FakeComponentPrefix: updates
Components: main bad

will create a Release file with

Codename: bla
Suite: foo
Components: updates/main updates/bad

in it, but otherwise nothing is changed, while

Codename: bla/updates
Suite: foo/updates
FakeComponentPrefix: updates
Components: updates/main updates/bad

will also create a Release file with

Codename: bla
Suite: foo
Components: updates/main updates/bad

but the packages will actually be in the components **updates/main** and **updates/bad**, most likely causing the same file using duplicate storage space.

This makes the distribution look more like Debian's security archive, thus work around problems with apt's workarounds for that.

AlsoAcceptFor
A list of distribution names. When a **.changes** file is told to be included into this distribution with the **include** command and the distribution header of that file is neither the codename, nor the suite name, nor any name from the list, a **wrongdistribution** error is generated. The **process_incoming** command will also use this field, see the description of **Allow** and **Default** from the **conf/incoming** file for more information.

Version
This optional field is simply copied into the Release files.

Origin This optional field is simply copied into the Release files.

Label This optional field is simply copied into the Release files.

NotAutomatic
This optional field is simply copied into the Release files. (The value is handled as an arbitrary string, though anything but **yes** does not make much sense right now.)

ButAutomaticUpgrades
This optional field is simply copied into the Release files. (The value is handled as an arbitrary string, though anything but **yes** does not make much sense right now.)

Description
This optional field is simply copied into the Release files.

Architectures
This required field lists the binary architectures within this distribution and if it contains **source** (i.e. if there is an item **source** in this line this Distribution has source. All other items specify things to be put after "binary−" to form directory names and be checked against "Architecture:" fields.)

This will also be copied into the Release files. (With exception of the **source** item, which will not occur in the topmost Release file whether it is present here or not)

Components
This required field lists the component of a distribution. See **GUESSING** for rules which component packages are included into by default. This will also be copied into the Release files.

UDebComponents
Components with a debian−installer subhierarchy containing .udebs. (E.g. simply "main")

Update
When this field is present, it describes which update rules are used for this distribution. There also can be a magic rule minus ("−"), see below.

Pull When this field is present, it describes which pull rules are used for this distribution. Pull rules are like Update rules, but get their stuff from other distributions and not from external sources. See the description for **conf/pulls**.

SignWith
When this field is present, a Release.gpg file will be generated. If the value is "yes" or "default", the default key of gpg is used. If the field starts with an exlamation mark ("!"), the given script is executed to do the signing. Otherwise the value will be given to libgpgme to determine to key to use.

If there are problems with signing, you can try
gpg −−list−secret−keys *value*
to see how gpg could interprete the value. If that command does not list any keys or multiple ones, try to find some other value (like the keyid), that gpg can more easily associate with a unique key.

If this key has a passphrase, you need to use gpg−agent or the insecure option **−−ask−passphrase**.

A '**!**' hook script is looked for in the confdir, unless it starts with **~/**, **./**, **+b/**, **+o/**, **+c/** or **/** . Is gets three command line arguments: The filename to sign, an empty argument or the filename to create with an inline signature (i.e. InRelease) and an empty argument or the filename to create an detached signature (i.e. Release.gpg). The script may generate no Release.gpg file if it choses to (then the repository will look like unsigned for older clients), but generating empty files is not allowed. Reprepro waits for the script to finish and will abort the exporting of the distribution this signing is part of unless the scripts returns normally with exit code 0. Using a space after ! is recommended to avoid incompatibilities with possible future extensions.

DebOverride
When this field is present, it describes the override file used when including .deb files.

UDebOverride
When this field is present, it describes the override file used when including .udeb files.

DscOverride
When this field is present, it describes the override file used when including .dsc files.

DebIndices, **UDebIndices**, **DscIndices**
Choose what kind of Index files to export. The first part describes what the Index file shall be called. The second argument determines the name of a Release file to generate or not to generate if missing. Then at least one of "**.**", "**.gz**", "**.xz**" or "**.bz2**" specifying whether to generate uncompressed output, gzipped output, bzip2ed output or any combination. (bzip2 is only available when compiled with bzip2 support, so it might not be available when you compiled it on your own, same for xz and liblzma). If an argument not starting with dot follows, it will be executed after all index files are generated. (See the examples for what argument this gets). The default is:
DebIndices: Packages Release . .gz
UDebIndices: Packages . .gz
DscIndices: Sources Release .gz

ExportOptions
Options to modify how and if exporting is done:
noexport Never export this distribution. That means there will be no directory below **dists/** generated and the distribution is only useful to copy packages to other distributions.
keepunknown Ignore unknown files and directories in the exported directory. This is currently the only available option and the default, but might change in the future, so it can already be requested explicitly.

Contents
Enable the creation of Contents files listing all the files within the binary packages of a distribution. (Which is quite slow, you have been warned).

In earlier versions, the first argument was a rate at which to extract file lists. As this did not work and was no longer easily possible after some factorisation, this is no longer supported.

The arguments of this field is a space separated list of options. If there is a **udebs** keyword, **.udeb**s are also listed (in a file called **uContents**−*architecture*.) If there is a**nodebs** k eyword, **.deb**s are not listed. (Only useful together with **udebs**) If there is at least one of the keywords **.**, **.gz**, **.xz** and/or **.bz2**, the Contents files are written uncompressed, gzipped and/or bzip2ed instead of only gzipped.

If there is a **percomponent** then one Contents−*arch* file per component is created. If there is a **allcomponents** then one global Contents−*arch* file is generated. If both are given, both are created. If none of both is specified then **percomponent** is taken as default (earlier versions had other defaults).

The switches **compatsymlink** or **nocompatsymlink** (only possible if **allcomponents** was not specified explicitly) control whether a compatibility symlink is created so old versions of apt−file

looking for the component independent filenames at least see the contents of the first component.

Unless **allcomponents** is given, **compatsymlinks** currently is the default, but that will change in some future (current estimate: after wheezy was released)

ContentsArchitectures
Limit generation of Contents files to the architectures given. If this field is not there, all architectures are processed. An empty field means no architectures are processed, thus not very useful.

ContentsComponents
Limit what components are processed for the **Contents**−*arch* files to the components given. If this field is not there, all components are processed. An empty field is equivalent to specify **nodebs** in the **Contents** field, while a non-empty field overrides a **nodebs** there.

ContentsUComponents
Limit what components are processed for the uContents files to the components given. If this field is not there and there is the **udebs** keyword in the Contents field, all .udebs of all components are put in the **uContents.***arch* files. If this field is not there and there is no **udebs** keyword in the Contents field, no **uContents**−*arch* files are generated at all. A non-empty fields implies generation of **uContents**−*arch* files (just like the **udebs** keyword in the Contents field), while an empty one causes no **uContents**−*arch* files to be generated.

Uploaders
Specifies a file (relative to confdir if not starting with **˜/**, **+b/**, **+c/** or **/**) to specify who is allowed to upload packages. Without this there are no limits, and this file can be ignored via **−−ignore=uploaders**. See the section**UPLO ADERS FILES** below.

Tracking
Enable the (experimental) tracking of source packages. The argument list needs to contain exactly one of the following:
keep Keeps all files of a given source package, until that is deleted explicitly via **removetrack**. This is currently the only possibility to keep older packages around when all indices contain newer files.
all Keep all files belonging to a given source package until the last file of it is no longer used within that distribution.
minimal Remove files no longer included in the tracked distribution. (Remove changes, logs and includebyhand files once no file is in any part of the distribution).
And any number of the following (or none):
includechanges Add the .changes file to the tracked files of a source package. Thus it is also put into the pool.
includebyhand Add **byhand** and **raw−*** files to the tracked files and thus in the pool.
includebuildinfos Add buildinfo files to the tracked files and thus in the pool.
includelogs Add log files to the tracked files and thus in the pool. (Not that putting log files in changes files is a reprepro extension not found in normal changes files)
embargoalls Not yet implemented.
keepsources Even when using minimal mode, do not remove source files until no file is needed any more.
needsources Not yet implemented.

Log Specify a file to log additions and removals of this distribution into and/or external scripts to call when something is added or removed. The rest of the**Log:** line is the filename, e very following line (as usual, have to begin with a single space) the name of a script to call. The name of the script may be preceded with options of the form **−−type=**(**dsc**|**deb**|**udeb**), **−−architecture=***name* or **−−component=***name* to only call the script for some parts of the distribution. An script with argument **−−changes** is called when a **.changes** file was accepted by **include** or **processincoming** (and with other arguments). Both type of scripts can have a **−−via=***command* specified, in which case it is only called when caused by reprepro command *command*.

For information how it is called and some examples take a look at manual.html in reprepro's source or **/usr/share/doc/reprepro/**

If the filename for the log files does not start with a slash, it is relative to the directory specified with **−−logdir**, the scripts are relative to **−−confdir** unless starting with **~/**, **+b/**, **+c/** or /.

ValidFor
If this field exists, an Valid−Until field is put into generated **Release** files for this distribution with an date as much in the future as the argument specifies.

The argument has to be an number followed by one of the units **d**, **m** or **y**, where **d** means days, **m** means 31 days and **y** means 365 days. So **ValidFor: 1m 11 d** causes the generation of a **Valid−Until:** header in Release files that points 42 days into the future.

ReadOnly
Disallow all modifications of this distribution or its directory in **dists/***codename* (with the exception of snapshot subdirectories).

ByHandHooks
This species hooks to call for handling byhand/raw files by processincoming (and in future versions perhaps by include).

Each line consists out of 4 arguments: A glob pattern for the section (clasically **byhand**, though Ubuntu uses **raw−***), a glob pattern for the priority (not usually used), and a glob pattern for the filename.

The 4th argument is the script to be called when all of the above match. It gets 5 arguments: the codename of the distribution, the section (usually **byhand**), the priority (usually only −), the filename in the changes file and the full filename (with processincoming in the secure TempDir).

Signed−By
This optional field is simply copied into the Release files. It is used to tell apt which keys to trust for this Release in the future. (see SignWith for how to tell reprepro whether and how to sign).

conf/updates

Name The name of this update−upstream as it can be used in the **Update** field in conf/distributions.

Method
An URI as one could also give it apt, e.g. *http://ftp.debian.de/debian* which is simply given to the corresponding **apt−get** method. (So either **apt−get has to be installed, or you have to point with −−methoddir** to a place where such methods are found.

Fallback
(Still experimental:) A fallback URI, where all files are tried that failed the first one. They are given to the same method as the previous URI (e.g. both http://), and the fallback-server must have everything at the same place. No recalculation is done, but single files are just retried from this location.

Config This can contain any number of lines, each in the format **apt−get −−option** would expect. (Multiple lines - as always - marked with leading spaces).

For example: Config: Acquire::Http::Proxy=http://proxy.yours.org:8080

From The name of another update rule this rules derives from. The rule containing the **From** may not contain **Method**, **Fallback** or **Config**. All other fields are used from the rule referenced in**Fr om**, unless found in this containing the **From**. The rule referenced in**Fr om** may itself contain a **From**. Reprepro will only assume two remote index files are the same, if both get their **Method** information from the same rule.

Suite The suite to update from. If this is not present, the codename of the distribution using this one is used. Also "*/whatever" is replaced by "<codename>/whatever"

Components
The components to update. Each item can be either the name of a component or a pair of a upstream component and a local component separated with ">". (e.g. "main>all contrib>all non−free>notall")

If this field is not there, all components from the distribution to update are tried.

An empty field means no source or .deb packages are updated by this rule, but only .udeb packages, if there are any.

A rule might list components not available in all distributions using this rule. In this case unknown components are silently ignored. (Unless you start reprepro with the **−−fast** option, it will warn about components unusable in all distributions using that rule. As exceptions, unusable components called **none** are never warned about, for compatibility with versions prior to 3.0.0 where and empty field had a different meaning.)

Architectures
The architectures to update. If omitted all from the distribution to update from. (As with components, you can use ">" to download from one architecture and add into another one. (This only determine in which Package list they land, it neither overwrites the Architecture line in its description, nor the one in the filename determined from this one. In other words, it is no really useful without additional filtering))

UDebComponents
Like **Components** but for the udebs.

VerifyRelease
Download the **Release.gpg** file and check if it is a signature of the **Releasefile** with the key given here. (In the Format as "gpg −−with−colons −−list−key" prints it, i.e. the last 16 hex digits of the fingerprint) Multiple keys can be specified by separating them with a "|" sign. Then finding a signature from one of the will suffice. To allow revoked or expired keys, add a "**!**" behind a key. (but to accept such signatures, the appropriate **−−ignore** is also needed). To also allow subkeys of a specified key, add a "+" behind a key.

IgnoreRelease: yes
If this is present, no **InRelease** or **Release** file will be downloaded and thus the md5sums of the other index files will not be checked.

GetInRelease: no
IF this is present, no **InRelease** file is downloaded but only **Release** (and **Release.gpg**) are tried.

Flat If this field is in an update rule, it is supposed to be a flat repository, i.e. a repository without a **dists** dir and no subdirectories for the index files. (If the corresponding **sources.list** line has the suite end with a slash, then you might need this one.) The argument for the **Flat:** field is the Component to put those packages into. No **Components** or **UDebComponents** fields are allowed in a flat update rule. If the **Architecture** field has any > items, the part left of the ">" is ignored.
For example the **sources.list** line
deb http://cran.r−project.org/bin/linux/debian etch−cran/
would translate to
Name: R
Method: http://cran.r−project.org/bin/linux/debian
Suite: etch−cran
Flat: whatevercomponentyoudlikethepackagesin

IgnoreHashes
This directive tells reprepro to not check the listed hashes in the downloaded Release file (and only in the Release file). Possible values are currently **md5**, **sha1** and **sha256**.

Note that this does not speed anything up in any measurable way. The only reason to specify this if

the Release file of the distribution you want to mirror from uses a faulty algorithm implementation. Otherwise you will gain nothing and only lose security.

FilterFormula
This can be a formula to specify which packages to accept from this source. The format is misusing the parser intended for Dependency lines. To get only architecture all packages use "architecture (== all)", to get only at least important packages use "priority (==required) | priority (==important)".

See the description of the listfilter command for the semantics of formulas.

FilterList, **FilterSrcList**
These two options each take at least two arguments: The first argument is the fallback (default) action. All following arguments are treated as file names of lists.

The filenames are considered to be relative to **−−confdir**, if not starting with **~/**, **+b/**, **+c/** or /.

Each list file consists of lines with a package name followed by whitespaced followed by an action.

Each list may only contain a single line for a given package name. The action to be taken is the action specified by the first file mentioning that package. If no list file mentions a package, the fallback action is used instead.

This format is inspired by dpkg −−get−selections before multiarch and the names of the actions likely only make sense if you imagine the file to be the output of this command of an existing system.

For each package available in the distribution to be updated from/pulled from this action is determined and affects the current decision what to do to the target distribution. (Only after all update/pull rules for a given target distribution have been processed something is actually done).

The possible action keywords are:

install mark the available package to be added to the target distribution unless the same version or a higher version is already marked as to be added/kept. (Note that without a prior delete rule (−) or **supersede** action, this will will never downgrade a package as the already existing version is marked to be kept).

upgradeonly
like **install** but will not add new packages to a distribution.

supersede
unless the current package version is higher than the available package version, mark the package to be deleted in the target distribution. (Useful to remove packages in add-on distributions once they reached the base distribution).

deinstall or **purge**
ignore the newly available package.

warning
print a warning message to stderr if a new package/newer version is available. Otherwise ignore the new package (like with **deinstall** or **purge**).

hold the new package is ignored, but every previous decision to downgrade or delete the package in the target distribution is reset.

error abort the whole upgrade/pull if a new package/newer version is available

= *version*

If the candidate package has the given version, behave like **install**. Otherwise continue as if this list file did not mention this package (i.e. look in the remaining list files or use the fallback action). Only one such entry per package is currently supported and the version is currently compared as string.

If there is both **FilterList** and **FilterSrcList** then the first is used for **.deb** and **.udeb** and the second for **.dsc** packages.

If there is only **FilterList** that is applied to everything.

If there is only **FilterSrcList** that is applied to everything, too, but the source package name (and source version) is used to do the lookup.

OmitExtraSourceOnly

This field controls whether source packages with Extra-Source-Only set are ignore when getting source packages. Withouth this option or if it is true, those source packages are ignored, while if set to no or false, those source packages are also condidates if no other filter excludes them. (The default of true will likely change once reprepro supports multiple versions of a package or has other means to keep the source packages around).

ListHook

If this is given, it is executed for all downloaded index files with the downloaded list as first and a filename that will be used instead of this. (e.g. "ListHook: /bin/cp" works but does nothing.)

If a file will be read multiple times, it is processed multiple times, with the environment variables **REPREPRO_FILTER_CODENAME**, **REPREPRO_FILTER_PACKAGETYPE**, **REPREPRO_FILTER_COMPONENT** and **REPREPRO_FILTER_ARCHITECTURE** set to the where this file will be added and **REPREPRO_FILTER_PATTERN** to the name of the update rule causing it.

ListShellHook

This is like ListHook, but the whole argument is given to the shell as argument, and the input and output file are stdin and stdout.

i.e.:
ListShellHook: cat
works but does nothing but useless use of a shell and cat, while
ListShellHook: grep−dctrl −X −S apt −o −X −S dpkg || [$? −eq 1]
will limit the update rule to packages from the specified source packages.

DownloadListsAs

The arguments of this field specify which index files reprepro will download.

Allowed values are **.**, **.gz**, **.bz2**, **.lzma**, **.xz**, **.diff**, **force.gz**, **force.bz2**, **force.lzma**, **force.xz**, and **force.diff**.

Reprepro will try the first supported variant in the list given: Only compressions compiled in or for which an uncompressor was found are used. Unless the value starts with **force.**, it is only tried if is found in the Release or InRelease file.

The default value is **.diff .xz .lzma .bz2 .gz .**, i.e. download Packages.diff if listed in the Release file, otherwise or if not usable download .xz if listed in the Release file and there is a way to uncompress it, then .lzma if usable, then .bz2 if usable, then .gz and then uncompressed).

Note there is no way to see if an uncompressed variant of the file is available (as the Release file always lists their checksums, even if not there), so putting '**.**' anywhere but as the last argument can mean trying to download a file that does not exist.

Together with **IgnoreRelease** reprepro will download the first in this list that could be unpacked (i.e. **force** is always assumed) and the default value is **.gz .bzip2 . .lzma .xz**.

conf/pulls
This file contains the rules for pulling packages from one distribution to another. While this can also be done with update rules using the file or copy method and using the exported indices of that other distribution, this way is faster. It also ensures the current files are used and no copies are made. (This also leads to the limitation that pulling from one component to another is not possible.)

Each rule consists out of the following fields:

Name The name of this pull rule as it can be used in the **Pull** field in conf/distributions.

From The codename of the distribution to pull packages from.

Components
The components of the distribution to get from.

If this field is not there, all components from the distribution to update are tried.

A rule might list components not available in all distributions using this rule. In this case unknown components are silently ignored. (Unless you start reprepro with the −−fast option, it will warn about components unusable in all distributions using that rule. As exception, unusable components called **none** are never warned about, for compatibility with versions prior to 3.0.0 where and empty field had a different meaning.)

Architectures
The architectures to update. If omitted all from the distribution to pull from. As in **conf/updates**, you can use ">" to download from one architecture and add into another one. (And again, only useful with filtering to avoid packages not architecture **all** to migrate).

UDebComponents
Like **Components** but for the udebs.

FilterFormula

FilterList

FilterSrcList
The same as with update rules.

OVERRIDE FILES

The format of override files used by reprepro should resemble the extended ftp−archive format, to be specific it is:

packagename field name new value

For example:
kernel−image−2.4.31−yourorga Section protected/base
kernel−image−2.4.31−yourorga Priority standard
kernel−image−2.4.31−yourorga Maintainer That's me <me@localhost>
reprepro Priority required

All fields of a given package will be replaced by the new value specified in the override file with the exception of special fields starting with a dollar sign ($). While the field name is compared case-insensitive, it is copied in exactly the form in the override file there. (Thus I suggest to keep to the exact case it is normally found in index files in case some other tool confuses them.) More than copied is the Section header (unless **−S** is supplied), which is also used to guess the component (unless **−C** is there).

Some values like **Package**, **Filename**, **Size** or **MD5sum** are forbidden, as their usage would severly

confuse reprepro.

As an extension reprepro also supports patterns instead of packagenames. If the package name contains '*', '[' or '?', it is considered a pattern and applied to each package that is not matched by any non-pattern override nor by any previous pattern.

Fieldnames starting with a dollar ($) are not be placed in the exported control data but have special meaning. Unknown ones are loudly ignored. Special fields are:

$Component: includedeb, includedsc, include and processincoming will put the package in the component given as value (unless itself overridden with **−C**). Note that the proper way to specify the component is by setting the section field and using this extension will most likely confuse people and/or tools.

$Delete: the value is treated a fieldname and fields of that name are removed. (This way one can remove fields previously added without removing and readding the package. And fields already included in the package can be removed, too).

conf/incoming

Every chunk is a rule set for the **process_incoming** command. Possible fields are:

Name The name of the rule-set, used as argument to the scan command to specify to use this rule.

IncomingDir

The Name of the directory to scan for **.changes** files.

TempDir

A directory where the files listed in the processed .changes files are copied into before they are read. You can avoid some copy operations by placing this directory within the same moint point the pool hierarchy is (at least partially) in.

LogDir

A directory where .changes files, .log files, .buildinfo files and otherwise unused .byhand files are stored upon procession.

Allow Each argument is either a pair *name1>name2* or simply *name* which is short for *name>name*. Each *name2* must identify a distribution, either by being Codename, a unique Suite, or a unique AlsoAcceptFor from **conf/distributions**. Each upload has each item in its**Distrib ution:** header compared first to last with each *name1* in the rules and is put in the first one accepting this package. e.g.:
Allow: local unstable>sid
or
Allow: stable>security−updates stable>proposed−updates
(Note that this makes only sense if Multiple is set to true or if there are people only allowed to upload to proposed−updates but not to security−updates).

Default *distribution*

Every upload not put into any other distribution because of an Allow argument is put into *distribution* if that accepts it.

Multiple

Old form of Options: multiple_distributions.

Options

A list of options
multiple_distributions
Allow including a upload in multiple distributions.

If a .changes file lists multiple distributions, then reprepro will start with the first name given, check all Accept and Default options till it finds a distribution this upload can go into.

If this found no distribution or if this option was given, reprepro will then do the same with the second distribution name given in the .changes file and so on.

limit_arch_all
If an upload contains binaries from some architecture and architecture all packages, the architecture all packages are only put into the architectures within this upload. Useful to combine with the **flood** command.

Permit A list of options to allow things otherwise causing errors:

unused_files
Do not stop with error if there are files listed in the **.changes** file if it lists files not belonging to any package in it.

older_version
Ignore a package not added because there already is a strictly newer version available instead of treating this as an error.

unlisted_binaries
Do not abort with an error if a .changes file contains .deb files that are not listed in the Binaries header.

Cleanup *options*
A list of options to cause more files in the incoming directory to be deleted:

unused_files
If there is **unused_files** in **Permit** then also delete those files when the package is deleted after successful processing.

unused_buildinfo_files
If .buildinfo files of processed .changes files are not used (neither stored by LogDir nor with Tracking: includebuildinfos) then delete them from the incoming dir. (This option has no additional effect if **unused_files** is already used.)

on_deny
If a **.changes** file is denied processing because of missing signatures or allowed distributions to be put in, delete it and all the files it references.

on_error
If a **.changes** file causes errors while processing, delete it and the files it references.

Note that allowing cleanup in publically accessible incoming queues allows a denial of service by sending in .changes files deleting other peoples files before they are completed. Especially when .changes files are handled directly (e.g. by inoticoming).

MorgueDir
If files are to be deleted by Cleanup, they are instead moved to a subdirectory of the directory given as value to this field. This directory has to be on the same partition as the incoming directory and files are moved (i.e. owner and permission stay the same) and never copied.

UPLOADERS FILES

These files specified by the **Uploaders** header in the distribution definition as explained above describe what key a **.changes** file as to be signed with to be included in that distribution.

Empty lines and lines starting with a hash are ignored, every other line must be of one of the following nine forms or an include directive:

allow *condition* **by anybody**
which allows everyone to upload packages matching *condition*,

allow *condition* **by unsigned**
which allows everything matching that has no pgp/gpg header,

allow *condition* **by any key**
which allows everything matching with any valid signature in or

allow *condition* **by key** *key-id*
which allows everything matching signed by this *key-id* (to be specified without any spaces). If the *key-id* ends with a **+** (plus), a signature with a subkey of this primary key also suffices.

key-id must be a suffix of the id libgpgme uses to identify this key, i.e. a number of hexdigits from the end of the fingerprint of the key, but no more than what libgpgme uses. (The maximal number should be what gpg −−list-key −−with−colons prints, as of the time of this writing that is at most 16 hex-digits).

allow *condition* **by group** *groupname*
which allows every member of group *groupname*. Groups can be manipulated by

group *groupname* **add** *key-id*
to add a *key-id* (see above for details) to this group, or

group *groupname* **contains** *groupname*
to add a whole group to a group.

To avoid warnings in incomplete config files there is also

group *groupname* **empty**
to declare a group has no members (avoids warnings that it is used without those) and

group *groupname* **unused**
to declare that a group is not yet used (avoid warnings that it is not used).

A line starting with **include** causes the rest of the line to be interpreted as filename, which is opened and processed before the rest of the file is processed.

The only conditions currently supported are:

***** which means any package,

source '*name***'**
which means any package with source *name*. ('*****', '**?**' and '**[**..**]**' are treated as in shell wildcards).

sections '*name***'**(|**'***name***'**)*
matches an upload in which each section matches one of the names given. As upload conditions are checked very early, this is the section listed in the .changes file, not the one from the override file. (But this might change in the future, if you have the need for the one or the other behavior, let me know).

sections contain '*name***'**(|**'***name***'**)*
The same, but not all sections must be from the given set, but at least one source or binary package needs to have one of those given.

binaries '*name***'**(|**'***name***'**)*
matches an upload in which each binary (type deb or udeb) matches one of the names given.

binaries contain '*name***'**(|**'***name***'**)*
again only at least one instead of all is required.

architectures '*architecture***'**(|**'***name***'**)*
matches an upload in which each package has only architectures from the given set. **sour ce** and **all** are treated as unique architectures. Wildcards are not allowed.

architectures contain '*architecture***'**(|**'***architecture***'**)*
again only at least one instead of all is required.

byhand
matches an upload with at least one byhand file (i.e. a file with section **byhand** or **raw-***something*).

byhand '*section***'('***section***')***
matches an upload with at least one byhand file and all byhand files having a section listed in the list of given section. (i.e. **byhand 'byhand'|'raw-*'** is currently is the same as **byhand**).

distribution '*codename***'**
which means any package when it is to be included in *codename*. As the uploaders file is given by distribution, this is only useful to reuse a complex uploaders file for multiple distributions.

Putting **not** in front of a condition, inverses it's meaning. For example
allow not source 'r*' by anybody
means anybody may upload packages which source name does not start with an 'r'.

Multiple conditions can be connected with **and** and **or**, with **or** binding stronger (but both weaker than **not**). That means
allow source 'r*' and source '*xxx' or source '*o' by anybody
is equivalent to
allow source 'r*xxx' by anybody
allow source 'r*o' by anybody

(Other conditions will follow once somebody tells me what restrictions are useful. Currently planned is only something for architectures).

ERROR IGNORING

With **--ignore** on the command line or an *ignore* line in the options file, the following type of errors can be ignored:

brokenold (hopefully never seen)
If there are errors parsing an installed version of package, do not error out, but assume it is older than anything else, has not files or no source name.

brokensignatures
If a .changes or .dsc file contains at least one invalid signature and no valid signature (not even expired or from an expired or revoked key), reprepro assumes the file got corrupted and refuses to use it unless this ignore directive is given.

brokenversioncmp (hopefully never seen)
If comparing old and new version fails, assume the new one is newer.

dscinbinnmu
If a .changes file has an explicit Source version that is different the to the version header of the file, than reprepro assumes it is binary non maintainer upload (NMU). In that case, source files are not permitted in .changes files processed by **include** or **processincoming**. Adding **--ignore=dscinbinnmu** allows it for the **include** command.

emptyfilenamepart (insecure)
Allow strings to be empty that are used to construct filenames. (like versions, architectures, ...)

extension
Allow one to **includedeb** files that do not end with **.deb**, to **includedsc** files not ending in **.dsc** and to **include** files not ending in **.changes**.

forbiddenchar (insecure)
Do not insist on Debian policy for package and source names and versions. Thus allowing all 7-bit characters but slashes (as they would break the file storage) and things syntactically active (spaces, underscores in filenames in .changes files, opening parentheses in source names of binary packages). To allow some 8-bit chars additionally, use **8bit** additionally.

8bit (more insecure)
: Allow 8-bit characters not looking like overlong UTF-8 sequences in filenames and things used as parts of filenames. Though it hopefully rejects overlong UTF-8 sequences, there might be other characters your filesystem confuses with special characters, thus creating filenames possibly equivalent to **/mirror/pool/main/../../../etc/shadow** (Which should be safe, as you do not run reprepro as root, do you?) or simply overwriting your conf/distributions file adding some commands in there. So do not use this if you are paranoid, unless you are paranoid enough to have checked the code of your libs, kernel and filesystems.

ignore (for forward compatibility)
: Ignore unknown ignore types given to −−*ignore*.

flatandnonflat (only suppresses a warning)
: Do not warn about a flat and a non-flat distribution from the same source with the same name when updating. (Hopefully never ever needed.)

malformedchunk (I hope you know what you do)
: Do not stop when finding a line not starting with a space but no colon(:) in it. These are otherwise rejected as they have no defined meaning.

missingfield (safe to ignore)
: Ignore missing fields in a .changes file that are only checked but not processed. Those include: Format, Date, Urgency, Maintainer, Description, Changes

missingfile (might be insecure)
: When including a .dsc file from a .changes file, try to get files needed but not listed in the .changes file (e.g. when someone forgot to specify −sa to dpkg−buildpackage) from the directory the .changes file is in instead of erroring out. (−−**delete** will not work with those files, though.)

spaceonlyline (I hope you know what you do)
: Allow lines containing only (but non-zero) spaces. As these do not separate chunks as thus will cause reprepro to behave unexpected, they cause error messages by default.

surprisingarch
: Do not reject a .changes file containing files for a architecture not listed in the Architecture-header within it.

surprisingbinary
: Do not reject a .changes file containing .deb files containing packages whose name is not listed in the "Binary:" header of that changes file.

undefinedtarget (hope you are not using the wrong db directory)
: Do not stop when the packages.db file contains databases for codename/packagetype/component/architectures combinations that are not listed in your distributions file.

  This allows you to temporarily remove some distribution from the config files, without having to remove the packages in it with the **clearvanished** command. You might even temporarily remove single architectures or components, though that might cause inconsistencies in some situations.

undefinedtracking (hope you are not using the wrong db directory)
: Do not stop when the tracking file contains databases for distributions that are not listed in your **distributions** file.

  This allows you to temporarily remove some distribution from the config files, without having to remove the packages in it with the **clearvanished** command. You might even temporarily disable tracking in some distribution, but that is likely to cause inconsistencies in there, if you do not know, what you are doing.

unknownfield (for forward compatibility)
: Ignore unknown fields in the config files, instead of refusing to run then.

unusedarch (safe to ignore)
No longer reject a .changes file containing no files for any of the architectures listed in the Architecture-header within it.

unusedoption
Do not complain about command line options not used by the specified action (like **−−architecture**).

uploaders
The include command will accept packages that would otherwise been rejected by the uploaders file.

wrongarchitecture (safe to ignore)
Do not warn about wrong "Architecture:" lines in downloaded Packages files. (Note that wrong Architectures are always ignored when getting stuff from flat repostories or importing stuff from one architecture to another).

wrongdistribution (safe to ignore)
Do not error out if a .changes file is to be placed in a distribution not listed in that files' Distributions: header.

wrongsourceversion
Do not reject a .changes file containing .deb files with a different opinion on what the version of the source package is.
(Note: reprepro only compares literally here, not by meaning.)

wrongversion
Do not reject a .changes file containing .dsc files with a different version.
(Note: reprepro only compares literally here, not by meaning.)

expiredkey (I hope you know what you do)
Accept signatures with expired keys. (Only if the expired key is explicitly requested).

expiredsignature (I hope you know what you do)
Accept expired signatures with expired keys. (Only if the key is explicitly requested).

revokedkey (I hope you know what you do)
Accept signatures with revoked keys. (Only if the revoked key is explicitly requested).

GUESSING

When including a binary or source package without explicitly declaring a component with **−C** it will take the first component with the name of the section, being prefix to the section, being suffix to the section or having the section as prefix or any. (In this order)

Thus having specified the components: "main non−free contrib non−US/main non−US/non−free non−US/contrib" should map e.g. "non−US" to "non−US/main" and "contrib/editors" to "contrib", while having only "main non−free and contrib" as components should map "non−US/contrib" to "contrib" and "non−US" to "main".

NOTE: Always specify main as the first component, if you want things to end up there.

NOTE: unlike in dak, non−US and non−us are different things...

NOMENCLATURE

Codename the primary identifier of a given distribution. This are normally things like **sarge**, **etch** or **sid**.

basename
the name of a file without any directory information.

byhand
Changes files can have files with section 'byhand' (Debian) or 'raw−' (Ubuntu). Those files are not packages but other data generated (usually together with packages) and then uploaded together

with this changes files.

With reprepro those can be stored in the pool next to their packages with tracking, put in some log directory when using processincoming, or given to an hook script (currently only possible with processincoming).

filekey the position relative to the outdir. (as found in "Filename:" in Packages.gz)

full filename
the position relative to /

architecture
The term like **sparc**, **i386**, **mips**, ... To refer to the source packages, **source** is sometimes also treated as architecture.

component
Things like **main**, **non−free** and **contrib** (by policy and some other programs also called section, reprepro follows the naming scheme of apt here.)

section Things like **base**, **interpreters**, **oldlibs** and **non−free/math** (by policy and some other programs also called subsections).

md5sum
The checksum of a file in the format "*<md5sum of file> <length of file>*"

Some note on updates

A version is not overwritten with the same version.

reprepro will never update a package with a version it already has. This would be equivalent to rebuilding the whole database with every single upgrade. To force the new same version in, remove it and then update. (If files of the packages changed without changing their name, make sure the file is no longer remembered by reprepro. Without **−−keepunreferencedfiled** and without errors while deleting it should already be forgotten, otherwise a **deleteunreferenced** or even some **__forget** might help.)

The magic delete rule ("−").

A minus as a single word in the **Update:** line of a distribution marks everything to be deleted. The mark causes later rules to get packages even if they have (strict) lower versions. The mark will get removed if a later rule sets the package on hold (hold is not yet implemented, in case you might wonder) or would get a package with the same version (Which it will not, see above). If the mark is still there at the end of the processing, the package will get removed.

Thus the line "Update: − *rules* " will cause all packages to be exactly the highest Version found in *rules*. The line "Update: *near* − *rules* " will do the same, except if it needs to download packages, it might download it from *near* except when too confused. (It will get too confused e.g. when *near* or *rules* have multiple versions of the package and the highest in *near* is not the first one in *rules*, as it never remember more than one possible spring for a package.

Warning: This rule applies to all type/component/architecture triplets of a distribution, not only those some other update rule applies to. (That means it will delete everything in those!)

ENVIRONMENT VARIABLES

Environment variables are always overwritten by command line options, but overwrite options set in the **options** file. (Even when the options file is obviously parsed after the environment variables as the environment may determine the place of the options file).

REPREPRO_BASE_DIR
The directory in this variable is used instead of the current directory, if no **−b** or **−−basedir** options are supplied.
It is also set in all hook scripts called by reprepro (relative to the current directory or absolute, depending on how reprepro got it).

REPREPRO_CONFIG_DIR
The directory in this variable is used when no −−**confdir** is supplied.
It is also set in all hook scripts called by reprepro (relative to the current directory or absolute, depending on how reprepro got it).

REPREPRO_OUT_DIR
This is not used, but only set in hook scripts called by reprepro to the directory in which the **pool** subdirectory resides (relative to the current directory or absolute, depending on how reprepro got it).

REPREPRO_DIST_DIR
This is not used, but only set in hook scripts called by reprepro to the **dists** directory (relative to the current directory or absolute, depending on how reprepro got it).

REPREPRO_LOG_DIR
This is not used, but only set in hook scripts called by reprepro to the value setable by −−**logdir**.

REPREPRO_CAUSING_COMMAND

REPREPRO_CAUSING_FILE
Those two environment variable are set (or unset) in **Log:** and **ByHandHooks:** scripts and hint what command and what file caused the hook to be called (if there is some).

REPREPRO_CAUSING_RULE
This environment variable is set (or unset) in **Log:** scripts and hint what update or pull rule caused this change.

REPREPRO_FROM
This environment variable is set (or unset) in **Log:** scripts and denotes what other distribution a package is copied from (with pull and copy commands).

REPREPRO_FILTER_ARCHITECTURE

REPREPRO_FILTER_CODENAME

REPREPRO_FILTER_COMPONENT

REPREPRO_FILTER_PACKAGETYPE

REPREPRO_FILTER_PATTERN
Set in **FilterList:** and **FilterSrcList:** scripts.

GNUPGHOME
Not used by reprepro directly. But reprepro uses libgpgme, which calls gpg for signing and verification of signatures. And your gpg will most likely use the content of this variable instead of "˜/.gnupg". Take a look at **gpg**(1) to be sure. You can also tell reprepro to set this with the −−**gnupghome** option.

GPG_TTY
When there is a gpg−agent running that does not have the passphrase cached yet, gpg will most likely try to start some pinentry program to get it. If that is pinentry−curses, that is likely to fail without this variable, because it cannot find a terminal to ask on. In this cases you might set this variable to something like the value of **$(tty)** or **$SSH_TTY** or anything else denoting a usable terminal. (You might also want to make sure you actually have a terminal available. With ssh you might need the −**t** option to get a terminal even when telling gpg to start a specific command).

By default, reprepro will set this variable to what the symbolic link **/proc/self/fd/0** points to, if stdin is a terminal, unless you told with −−**noguessgpgtty** to not do so.

BUGS

Increased verbosity always shows those things one does not want to know. (Though this might be inevitable and a corollary to Murphy)

Reprepro uses berkeley db, which was a big mistake. The most annoying problem not yet worked around is database corruption when the disk runs out of space. (Luckily if it happens while downloading packages while updating, only the files database is affected, which is easy (though time consuming) to rebuild, see **recovery** file in the documentation). Ideally put the database on another partition to avoid that.

While the source part is mostly considered as the architecture **source** some parts may still not use this notation.

WORK-AROUNDS TO COMMON PROBLEMS

gpgme returned an impossible condition

With the woody version this normally meant that there was no .gnupg directory in $HOME, but it created one and reprepro succeeds when called again with the same command. Since sarge the problem sometimes shows up, too. But it is no longer reproducible and it does not fix itself, neither. Try running **gpg −−verify** *file-you-had-problems-with* manually as the user reprepro is running and with the same $HOME. This alone might fix the problem. It should not print any messages except perhaps
gpg: no valid OpenPGP data found.
gpg: the signature could not be verified.
if it was an unsigned file.

not including .orig.tar.gz when a .changes file's version does not end in −0 or −1

If dpkg−buildpackage is run without the **−sa** option to build a version with a Debian revision not being −0 or −1, it does not list the **.orig.tar.gz** file in the **.changes** file. If you want to **include** such a file with reprepro when the .orig.tar.gz file does not already exist in the pool, reprepro will report an error. This can be worked around by:
call **dpkg−buildpackage** with **−sa** (recommended)
copy the .orig.tar.gz file to the proper place in the pool before
call reprepro with −−ignore=missingfile (discouraged)

leftover files in the pool directory.

reprepro is sometimes a bit too timid of deleting stuff. When things go wrong and there have been errors it sometimes just leaves everything where it is. To see what files reprepro remembers to be in your pool directory but does not know anything needing them right know, you can use
reprepro dumpunreferenced
To delete them:
reprepro deleteunreferenced

INTERRUPTING

Interrupting reprepro has its problems. Some things (like speaking with apt methods, database stuff) can cause problems when interrupted at the wrong time. Then there are design problems of the code making it hard to distinguish if the current state is dangerous or non-dangerous to interrupt. Thus if reprepro receives a signal normally sent to tell a process to terminate itself softly, it continues its operation, but does not start any new operations. (I.e. it will not tell the apt−methods any new file to download, it will not replace a package in a target, unless it already had started with it, it will not delete any files gotten dereferenced, and so on).

It only catches the first signal of each type. The second signal of a given type will terminate reprepro. You will risk database corruption and have to remove the lockfile manually.

Also note that even normal interruption leads to code-paths mostly untested and thus expose a multitude of bugs including those leading to data corruption. Better think a second more before issuing a command than risking the need for interruption.

REPORTING BUGS

Report bugs or wishlist requests to the Debian BTS
(e.g. by using **reportbug reprepro** under Debian)
or directly to brlink@debian.org

COPYRIGHT