# BAR – Backup ARchiver

Version 0.13

# Table of Contents

## Illustration Index



## Index of Tables

# Overview

BAR is backup archive program to store files and disk images into archive files. It is working like other popular archive tools like *tar*, but offer a lot of useful features.

## *Features*

- create, list, test, compare and extract archives of files and disk images,
- fast access to entries in archives,
- full and incremental backup of file archives,
- support for raw and several file systems for disk images (ext, fat, reiserfs),
- split archives into parts of selectable size,
- compress archives with zip, bzip2 or lzma algorithms,
- encrypt archive content with gcrypt functions (BLOWFISH, TWOFISH, AES, a. o.),
- asymmetric encryption of content with RSA keys,
- store archives directly on external server via ftp or scp,
- with external tools: store archives on CD/DVD including error correction codes,
- server mode with included scheduler for doing backups regularly. Controlling the server can be done via network connection (plain & TLS/SSL),
- graphical front end for server to check server status, create jobs, start jobs and stop jobs.

## *Requirements*

To execute BAR you need:

- glibc 2.3.2 or higher
- BAR binary
- JRE 1.6 [JRE, JDK]

Optional external tools for creating CD/DVD with BAR:

- growisofs [growisofs]
- mkisofs [dvd+rw-tools]
- dvdisaster [dvdisaster]

BAR is either available as a binary package for different systems or as source code for compilation. If you are not familiar with using a C compiler it is recommended to use one of the binary packages. Please check the website http://www.kigen.de/projects/bar/index.html for available binary packages.

To compile BAR you need:

- C development environment with gcc, GNU make
- perl
- zlib library [zlib]
- JDK 1.6 [JRE, JDK]

Optional for compilation are:

- bzip2 library [bzip2]
- lzma library [LZMA]

- ssh2 library [libssh2]
- gcrypt library [libgcrypt]
- gnutls library [gnutls]
- ftplib [ftplib]
- ant, launch4j [ant, launch4j]
- epm [epm]

For more details to compile BAR by yourself please see chapter Compilation.

# Installation

## *Packages*

Installation of BAR can be done either from pre-compiled binary package files or from the sources. Pre-compiled binary packages are available from the web site for some major Linux distributions. If there is no pre-compiled binary package available for your platform, you must compile BAR by yourself from the sources. The chapter Compilation give some instructions how to compile BAR.

Installation of Debian packages:

*sudo dpkg –install <BAR package file>.deb*

Installation of RPM packages:

*sudo rpm –install <BAR package file>.deb*

## *Create key files*

To use the SSL network connection to the BAR daemon a SSL key-pair must be generated. The script file bar-keygen can be used to create he required public and private key files as well as the Java key file required for the frontend. To generate a new key-pair call:

*sudo bar-keygen*

This generate and install the key files in the appropriated directories shown in the following table:

| File | Description |
|---|---|
| /etc/ssl/cert/bar-ca.pem | BAR certificate |
| /etc/ssl/private/bar-key.pem | BAR private key |
| /etc/ssl/cert/bar-server-cert.pem | BAR server certificate |
| /etc/ssl/private/bar-server-key.pem | BAR server private key |
| /etc/bar/bar.jks | BAR Java front end key |

*Table 1: Certificates and key files*

Note: keep the key files always in private!

# Usage

The following chapters use the following nomenclature:

- <name> or <number> stay for a parameter, e. g. a name, a number or some text,
- [...] optional parameter,
- ... repeat the parameter one or more times ,
- <size> specify a byte size. Sizes can be specified as numbers including the following optional units: G, M, K which multiply the given number by 1073741824 ($1024^3$), 1048576 ($1024^2$), 1024 respectively.

## *Creating archives*

BAR can create archives containing files or block device images. With the options *-#* or *--include* resp. *-!* or *--exclude* pattern can be specified to explicitly include or exclude files or devices. By default include/exclude patterns can contain the meta characters * and *?* to match multiple or single characters. With the option *--pattern-type* also regular expression or extended regular expression patterns can be enabled.

Note: If * or *?* are used the pattern may be wrapped with '…' to avoid expansion of the pattern by the shell.

## File archives

To create an archive containing files enter the command:

*bar --create <archive file name> <file or directory>...*
*bar -c <archive file name> <file or directory>...*

Examples:

*bar --create home.bar /home*
*bar -c backup1.bar /home/foo/file1.txt /home/foo/file2.txt*
*bar -c backup2.bar /home/foo/*.txt -# 'documents/*' -! '*/trash/*'*

## Image archives

To create an archive containing the image of a disk or partition enter the command:

*bar --image <archive file name> <device name>...*
*bar -m <archive file name> <device name>...*

Examples:

*bar --image home.bar /dev/hda*
*bar -m home.bar /dev/hda1*

When you create an archive with images BAR tries to detect the used file system on the device. If the file system is know only blocks used by the file system are stored. Not used blocks are stored as “0”-blocks, thus when compression is used disk images get smaller. To disable detection of the file system and store all blocks (ram image) use the option *--raw-images*.

Currently supported file systems are:

- ext2
- ext3
- fat12
- fat16
- fat32
- reiserFS (version 3.5/3.6)

Experimental supported file systems are:

- ext4
- reiser4

## List archives content

To list the content of archives enter the command:

*bar --list <bar archive file>...*
*bar -l <bar archive file>...*

or simply:

*bar <bar archive file>...*

Examples:

*bar --list home.bar*
*bar -l backup1.bar backup2.bar*
*bar backup3.bar*

The content of the specified archives are listed in short list format with byte sizes. To get a long list format use the option *-L* or *--long-format*, to get human readable sizes use option *-H* or *--human-format.*

## Test archives

To test the integrity of an archive enter the following command:

*bar --test <bar archive file>...*
*bar -t <bar archive file>...*

Examples:

*bar --test home.bar*
*bar -t backup1.bar backup2.bar*

The specified archives are tested if all data can be read, decrypted and decompressed. If some data cannot be read, decrypted or decompressed an error is reported.

## Compare archives

To compare the content of archives with files, disk or partition images enter the following command:

*bar --compare <bar achive file>...*

*bar -d <bar achive file>...*

BAR compare the archive with the content on disk and report any file or image which differs. The compare function can be used to check if an archive still contain the files or images which are on disk.

Examples:

*bar --compare home.bar*
*bar -d backup1.bar backup2.bar*

The content of the specified archives are compared with the existing files or disk image content. If some file or image in an archive differ from the file/image on disk a message is printed.

## Extract archives

To extract (restore) the content of an archive enter the following command:

*bar --extract <bar archive file>...*
*bar -x <bar archive file>...*

BAR extract the content of the archive and restore all files and images which are stored in the specified archive files. Existing files will not be overwritten if not specified different by the option *--overwrite-files*. The destination of files/images can be modified with the option *--destination*.

Examples:

*bar --extract home.bar*
*bar -x backup1.bar backup2.bar*

## Compress archives

To save space in the created archive files it is highly recommended to compress the data. BAR offer different compression algorithms. If compression is enabled each entry in the archive is compressed separately. This result in a slightly bigger archive file than compressing the complete archive like e.. g. *tar* do with the appropriated compression algorithm, but with single compressed entries BAR can list and search archive content very fast, because it is not required to decompress the whole archive content before a single entry in the archive can be read. With separately compressed archive entries also not corrupted parts of a partially corrupted archive may still be readable.

To compress the archive content use the following option:

*--compress-algorithm=<name>*

or

*-z <name>*

The supported compression algorithm names can be listed with the option *-h* or --help. All possible compression algorithm names are listed in table 4: Compression algorithms (see page 19).

Examples:

*bar --create home.bar /home --compress-algorithm=zip1*
*bar --create home.bar /home -z bzip9*
*bar --create home.bar /home -z lzma9*

The resulting compression ratio depend on the used compression algorithm.

## Encrypt archives

BAR archive files can be encrypted. Encryption is done after compression and is highly recommended for archive files which are stored on external servers or Cds/DVDs to avoid non authorized access of the content.

To enable encryption use the following option:

*--crypt-algorithm=<name>*

or

*-y <name>*

When encryption is enabled, a pass phrase must be entered when the archive is created. The content of the archive may later only be accessed again when the pass phrase is entered correctly again.

Note: Because of the internal structure of the archive files, the general structure (how many entries, entry types) can be read without knowledge of the pass phrase. Nevertheless all data content are encrypted and can only be read with the correct pass phrase.

For encryption the crypt algorithms of libgcrypt [libgcrypt] are used. The supported crypt algorithm names can be listed with the option -h or --help. All possible crypt algorithm names are listed in table 5: Encryption algorithms (see page 20). The strength of encryption depend on the used algorithm and the used pass phrase. Currently e. g. AES256 may be a good choice for the encryption algorithm.

Note: Do not weaken the encryption by weak pass phrases! If your pass phase used for encryption is to weak and may be guessed easily encryption become useless.

Note: There is no way to restore the archive content if you forgot the pass phrase.

## Split archives

Complete backup archives can grow to big files depending on the content to store and the used compression algorithm. BAR offer the possibility to split archives into parts of a specific size. Parts can usually be stored more easily e. g. on a server or on CD/DVD. Each created part can read independently. Even when a single part is missing or cannot be read anymore because it became destroyed by some reason, e. g. read error on CD/DVD, other parts can still be read by BAR and restored. Thus not the whole archive will get lost. Parts which cannot be read are restored as “0”-bytes in the resulting file or image.

To split an archive to a specific size use the following option:

*--archive-part-size=<size>*

or

*-s <size>*

Note: The size of a single part may become a little bit larger than the specified size when compression is used. The reason for this is that the compression algorithms usually use an internal buffer for some data which may be flushed and appended to the archive part when the part size reached the specified size.

## *Store archives*

Created archives can be stored to several destination places by specifying an URI (Uniform Resource Identifier) at the beginning of an archive file name. The default is storing into the local file system when no URI is given. BAR can store archives directly to a remote server or with some externals tools also on CD/DVD.

To create an archive which should be stored at a specific destination use the following command:

*bar --create <uri><archive file name> <file or directory>...*

<uri> can be one of the URIs described in the following sections.

## File system

To store the archive into the file system of the local computer use the following URI:

*file:///<file name>*

or simply

*<file name>*

Examples:

*bar --create file:///backup/home.bar /home*
*bar --create /backup/home.bar /home*

## Remote server

BAR can store archives directly on a server either via the ftp protocol when the ftplib [ftplib] is compiled-in or via the ssh/scp protocol when the libssh2 [libssh2] is compiled-in.

To store the archive directly on a remote server use the following URI:

*ftp:///[<login name>[@<login password>]]<file name>*

*scp:///[<login name>[@<login password>]]<file name>*

The archive (part) is first created on the local hard disk in a temporary file and is then transmitted to the remote server. Thus the local hard disk should have enough free space to hold at least one archive part. See option *--max-tmp-size* to limit the space BAR may use for temporary files. Transmission of an archive part to a remove server is done with full speed of the network connection uplink. To limited the used bandwidth use the option *--max-band-with*.

Examples:

*bar --create ftp:///backup/home.bar /home*
*bar --create scp://foo@mypassword/backup/home.bar /home*

## CD/DVD

When the external tools *growisofs*, *mkisofs* and *dvdisaster [dvdisaster]* are installed, BAR can also create CDs and DVDs. To store the archives on CD or DVD use the following URI:

*dvd://<file name>*

To add error correction codes in the space not used on the CD/DVD the external tool *dvdisaster* can be used to add Reed Solomo error correction codes. BAR will call *dvdisaster* after creating the CD/DVD image when you specify the following option:

*--ecc*

Note: To create a CD or DVD the archive files are first created in a temporary directory before they are written to the CD/DVD. Thus the local hard disk should have enough free disk space to store the arachive files and additionally the size of the CD/DVD image. In general around 9GB of temporary disk space may be required. BAR will output a warning when the free disk space in the temporary directory may not be sufficient to create a CD or DVD.

Examples:

*bar --create dvd://home.bar /home*
*bar –ecc --create dvd://home.bar /home*

## Device

TODO

Example:

*bar --create device:///device/backup/home.bar /home*

## *Incremental archives*

BAR can also create incremental files archives (Note: not useful for images). For this BAR create an incremental data file which is used to detect which files changed and must be stored in an incremental archive. The incremental data files have by default the extension *.bid*. The full name is derived from the created archive file by the following pattern:

*<path>/<archive name><date><number>.bar*

will create an incremental data file with the name

*<path>/<archive name>.bid*

The derived default incremental data file name may be overwritten with the option *-I* or *–incremental-file-name*.

Note: Incremental data files are not encrypted, always stored on the local hard disk and are never transmitted to a remote server. The incremental data files are only used by BAR itself to create incremental archives.

To create an incremental archive first create an archive with all files using the following command:

*bar --full --incremental-file-name=<incremental data file name> --create <archive file name> <file or directory>...*

Example:

*bar --full --incremental-file-name=home.bid --create home.bar /home*

This example command create the file *home.bar* and *home.bid.*

To create the incremental archive after some files were modified use the command:

*bar --incremental --create <archive file name> <file or directory>...*
*bar -i --create <archive file name> <file or directory>...*

Example:

*bar --incremental --create --incremental-file-name=home.bid home-incremental.bar /home*

This command read in the incremental data file *home.bid* and store all modified files from /*home* in the new archive *home-incremental.bar*. When the incremental archive is created the incremetal data file *home.bid* is updated.

# Server

## Daemon

BAR can be started as a server (daemon) which is running in the background and execute backup or restore jobs. To control the server you can connect to a running BAR server instance on some computer – including on a remote system[1] – with a network connection. The network connection can either be a plain TCP/IP socket connection and it can be a protected and encrypted TLS (SSL)-connection (recommended).

To start BAR in server mode enter the following command:

*bar --daemon*

This start BAR and detach the process to run in the background.

BAR server use the default ports for accepting network connections from the graphical front end listed in table 2. The default port settings can be overwritten with the options shown in the table in the last column.

| Port | Type | Option to overwrite default value |
|---|---|---|
| 38523 | plain connection | --server-port |
| 38524 | TLS (SSL) connection | --tls-server-port |

*Table 2: Server ports*

## Graphical front end

The graphic front end for BAR can be used to control a BAR server instance running on the local or some remote computer. Via the front end the status of a server can be display, backup or restore jobs can be create, edited, deleted, started, paused, suspended or canceled.

Note: To execute the graphical front end the Java Run time Environment 1.6 [JRE, JDK] must be installed. The front end use SWT [SWT]. The required SWT libraries for Linux (32- and 64bit), Solaris (32bit), Windows (32- and 64bit), and OS X (32bit) are included in BAR.

The graphical front end is started with:

*barcontrol [<server>]*

The frontend try to connect to the specified server or the local computer if no sever name is

1 Accessing BAR on a remove system may require to open the BAR server control ports in routers and firewall settings.

specified either via a TLS/SSL encrypted connection or via a plain connection.

Note: For TLS/SSL encrypted connections a TLS/SSL key pair must be generated. The BAR tool *bar-keygen* can generate the key files. For this the key generator tools *certtool* from GNU TLS project [gnutls] or *openssl* from the OpenSSL project [OpenSSL] and the Java JRE tool *keytool* [JRE, JDK] must be installed on the computer.

The main window look like in example figure 1.

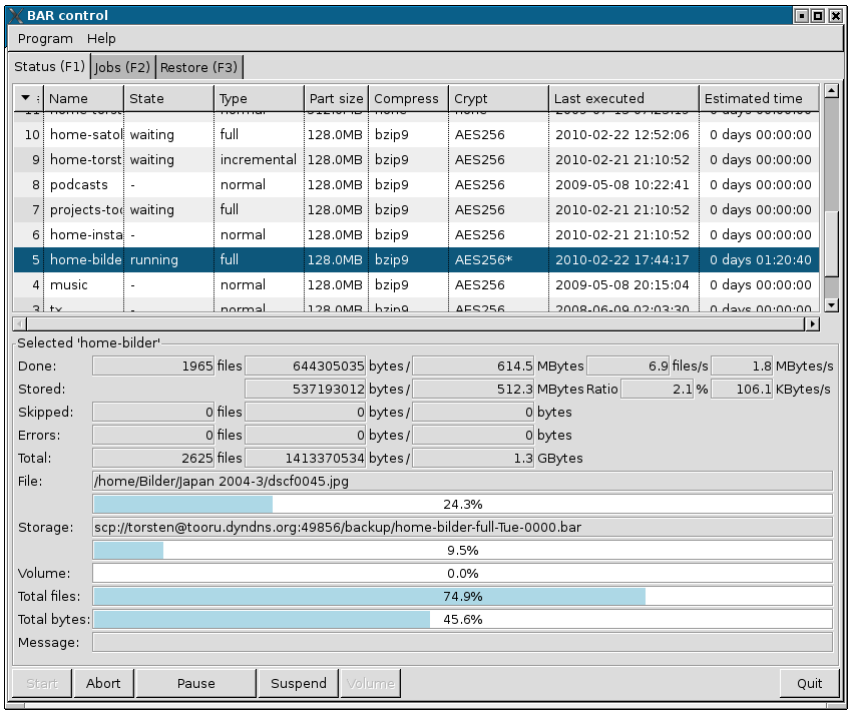


*Figure 1: Graphical front end main window (status tab)*

## Status tab

In the status tab (see example in figure 1.) the list in the upper part show the defined backup jobs and their current status. The lower part show the detailed status of the currently select job in the list. Via the button ruler at the bottom a backup job can be started, aborted, paused or suspend.

When data is written to a CD, a DVD or a device and a new volume is required, a message appear that a new volume should be inserted. With the button *Volume* BAR is informed that a new volume is loaded and can be used to store the next part of the data.

## Jobs tab

In the jobs tab backup jobs can be created, edited and deleted. Each job have a unique name and is stored into a text file by the BAR server (default directory is */etc/bar/jobs*). A job file contain the following data:

- file names
- images
- excluded files
- storage information
- schedule information

The jobs tab have some sub tabs for to edit these pieces of data.

### Files

With the files tab (see example figure 2) files and directories can be included and excluded into/from a backup job. A double-click on a directory open or close the directory. With the buttons at the bottom files, links, directories and special files can be included (entry become green) or excluded (entry has a red cross). Entries which are not either explicitly included or excluded will become included or excluded in the backup when the parent entry is included or excluded.

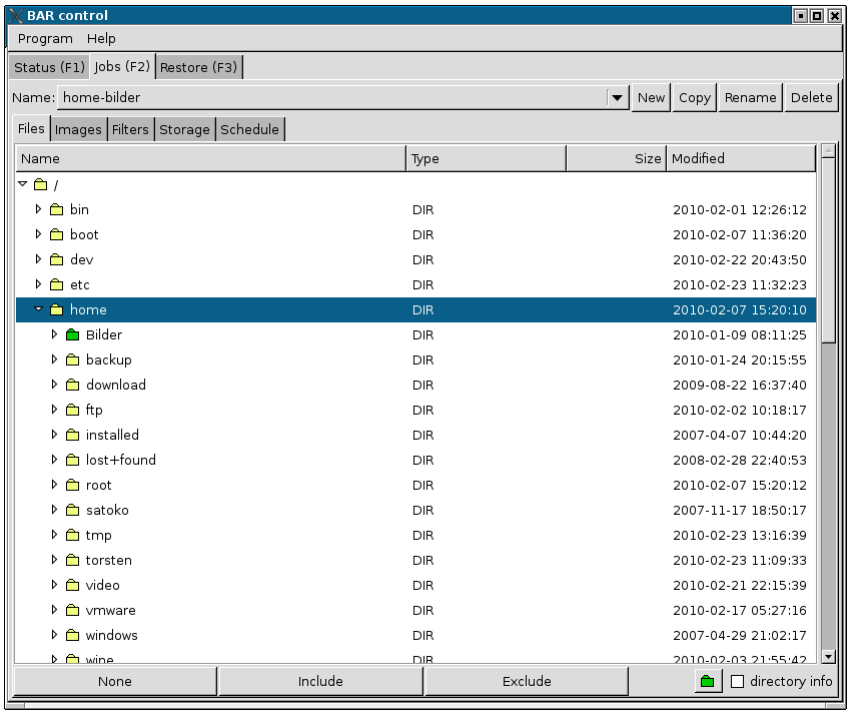


*Figure 2: Files*

### Images

Similar to files, links and directories in the images tab (see example figure 3) devices can be selected to be stored into an archive as block device image files.

Note: The block device should not be mounted while creating an image backup.

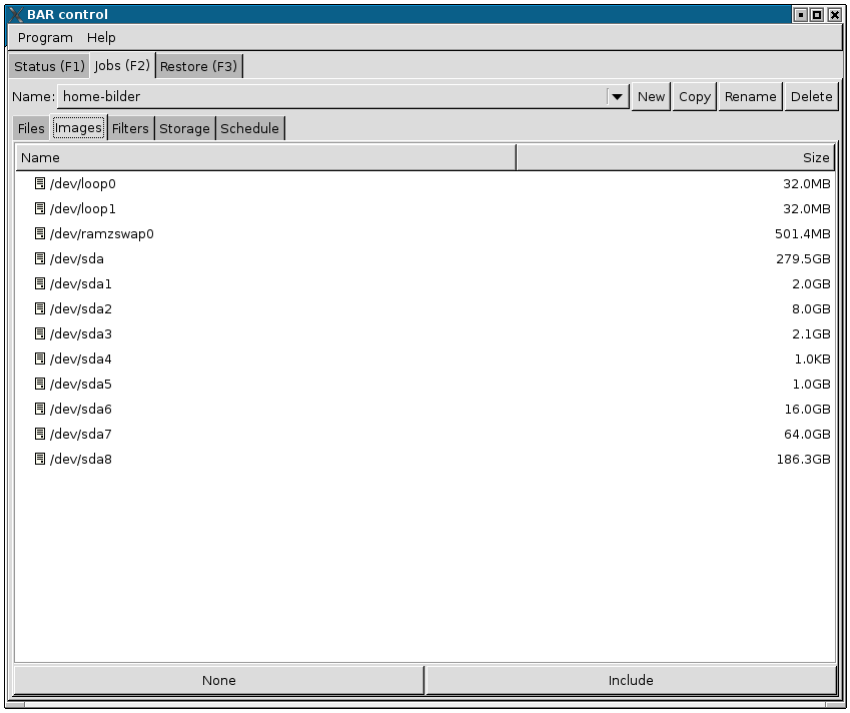


*Figure 3: Images*

### Filters

In the filters tab (see example figure 4) general include and exclude filters can be specified. Files and directories matching to an include filter and do not match to exclude filter are included into the created archive. Files and directories which are explicitly excluded will never be stored into an archive.

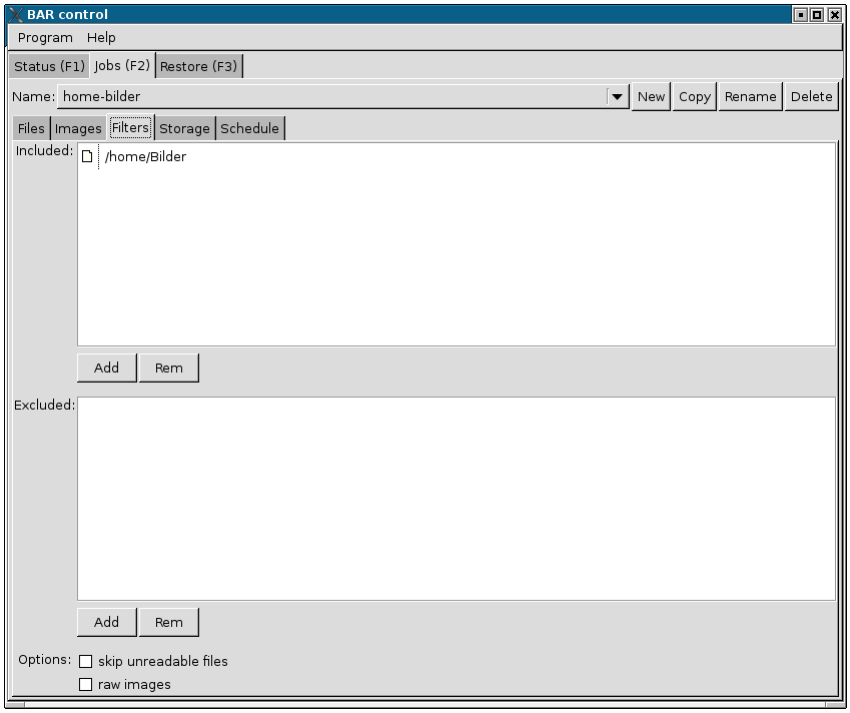


*Figure 4: Filters*

### Storage

With the storage tab (see example figure 5) different settings for storing created archive files can be edited. In this tab the archive part size, compression, encryption and the storage name and location

can be set.

In the archive file name field the button 🗀 to the right open an editor dialog for the archive file name. This dialog include all possible variable patterns the name may include (see example figure 6). The archive file name can simply be created by drag and drop of the variable patterns into the name field.

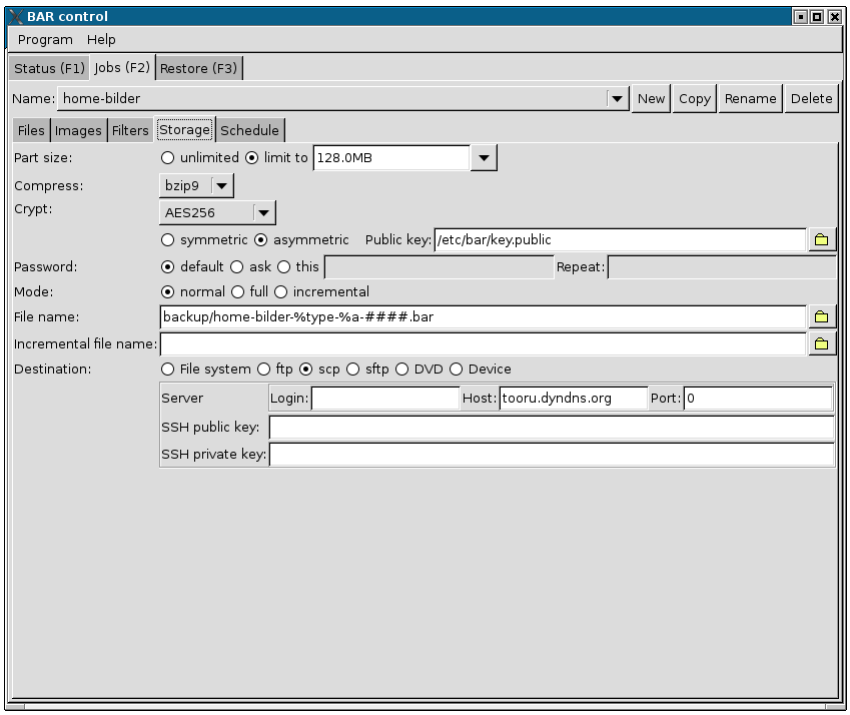


*Figure 5: Storage*

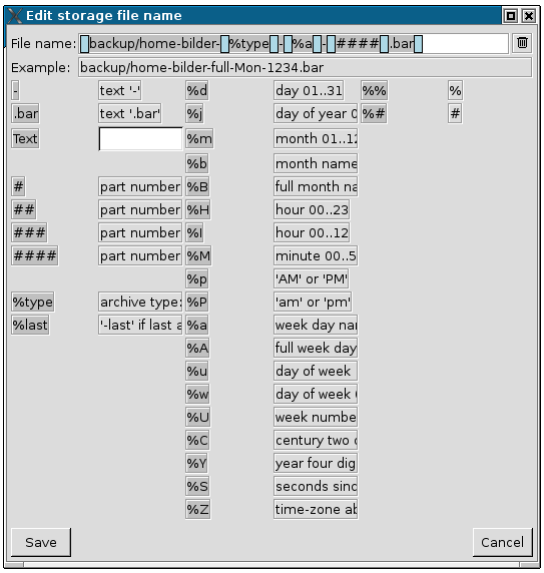


*Figure 6: Archive name editor*

### *Schedule*

In the schedule tab (see example figure 7) repeat schedule time and dates for the backup job can be defined. The backup job will automatically executed at the specified time and dates.

Note: If a backup job was missed to execute because the BAR server was not running at the specified time, the missed job will executed immediately when the BAR server is restarted

the next time.

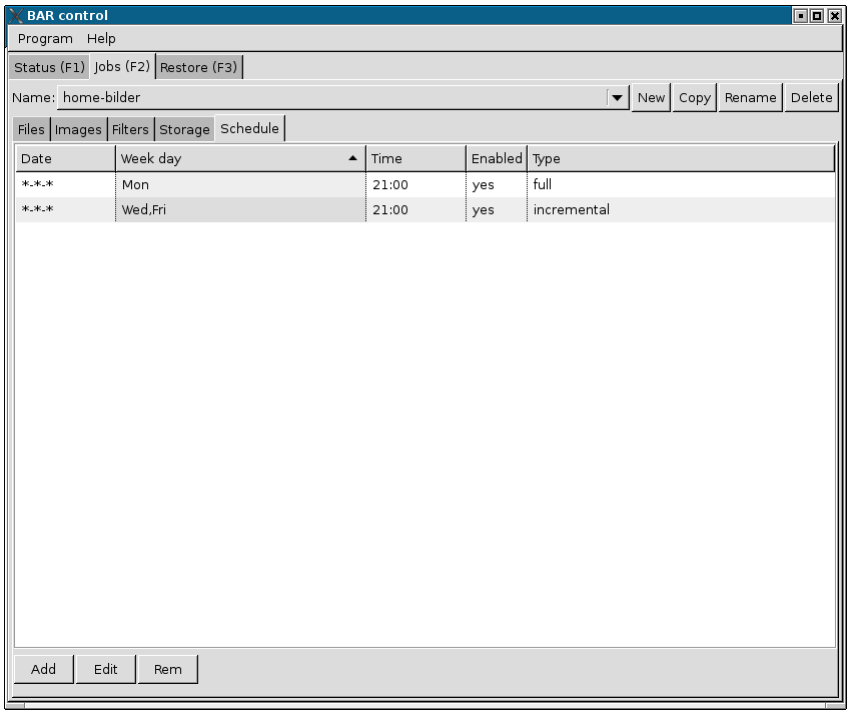


*Figure 7: Scheduler*

## Restore tab

TODO

# Appendix

## *Command line options summary*

## Basic commands

--create, -c

Create an archive with files, directories, links and special files like device-, pipes- and sockets-descriptors.

--image, -m

Create an archive with disk or partition images. BAR try to detect the file system on disk or partition and will store not used data blocks with binary content “0”. To disable this feature use option --*raw-images*.

Note: When compression is used not used data blocks will almost use no space in the created archive.

--list, -l

List content of archives.

--test, -t

Test content integrity of archives.

--compare, -d

Compare content of archives with content in file system.

--generate-keys

Generate a public/private key pair for asymmetric encryption/decryption of archive content with RSA.

Note: RSA encryption is only used for the public/private key pair. The archive content itself is encrypted with AES 256 with a random selected password key of 2048 bit. The random password is encrypted with the public key and stored into the archive. For each part a new random password is generated. This mechanism is know as a hybrid-encryption because plain RSA encryption of all data

--generate-keys-bits=<n>

Specify number of bits to use for RSA key pair. Default value are 2048 bits.

## Include/Exclude files

--include=<pattern>, -#

Define a simple file pattern (glob pattern), a regular expression, or a extended regular expression which must match to files which are included into the created archive. Default pattern type is simple file pattern. To select regular expression or extended regular expression pattern use one of the prefixes listed in the table 3.

| Prefix | Type | Example |
|---|---|---|
| g: | simple pattern (glob pattern) | g:/home/foo/*.txt |
| r: | regular expression | r:/home/foo/.*.txt |
| x: | extended regular expression | x:/home/foo/.*.txt |

*Table 3: Pattern types*

--exclude=<pattern>, -!

Define a simple file pattern (glob pattern), a regular expression, or a extended regular expression which must match to files which are excluded from the created archive. Default pattern type is simple file pattern. See table 3 how to specify different pattern types.

Note: Exclusions have precedence over include patterns.

## Compression

--compress-algorithm, -z

Specify the compression algorithm to use for archive content data. Which compression algorithm can be used depend on the compiled-in libraries. If all supported compression algorithms are

compiled into BAR then either none or one of the zip [zlib], bzip2 [bzip2] or LZMA [LZMA] algorithms can be used. Usually LZMA have a better compression ratio than BZIP2 than ZIP. The ratio depend on the size and type of data which should be compressed. Short files and already compressed files can usually not further compressed.

The different compression algorithms over different levels of compression. Usually a higher level result in a better compression, but require more cpu time. Table 4 list all supported compression algorithms.

| Name | Description |
|---|---|
| none | no compression |
| zip1 | ZIP compression level 1 |
| zip2 | ZIP compression level 2 |
| zip3 | ZIP compression level 3 |
| zip4 | ZIP compression level 4 |
| zip5 | ZIP compression level 5 |
| zip6 | ZIP compression level 6 |
| zip7 | ZIP compression level 7 |
| zip8 | ZIP compression level 8 |
| zip9 | ZIP compression level 9 |
| bzip1 | BZIP2 compression level 1 |
| bzip2 | BZIP2 compression level 2 |
| bzip3 | BZIP2 compression level 3 |
| bzip4 | BZIP2 compression level 4 |
| bzip5 | BZIP2 compression level 5 |
| bzip6 | BZIP2 compression level 6 |
| bzip7 | BZIP2 compression level 7 |
| bzip8 | BZIP2 compression level 8 |
| bzip9 | BZIP2 compression level 9 |
| lzma1 | LZMA compression level 1 |
| lzma2 | LZMA compression level 2 |
| lzma3 | LZMA compression level 3 |
| lzma4 | LZMA compression level 4 |
| lzma5 | LZMA compression level 5 |
| lzma6 | LZMA compression level 6 |
| lzma7 | LZMA compression level 7 |
| lzma8 | LZMA compression level 8 |
| lzma9 | LZMA compression level 9 |

*Table 4: Compression algorithms*

Note: In the archive files created by BAR each entry is compressed separately. This may result in slightly bigger files than archives which are compressed at whole, because meta data like the archive structuring data is not compressed as well compression of the separated files may be not as efficient than a single block of files. Nevertheless when each entry is compressed separately, each entry can be read and extracted separately as well, thus result in faster listing and access of single entries in an archive file.

--compress-min-size=<size>

Specify the minimal size in bytes a file must have to become compressed. Usually small files cannot be compressed efficiently. Thus it is better to store them directly into the archive.

## Encryption

--crypt-algorithm, -y

Specify the encryption algorithm to use for archive content data. Which encryption algorithm can be used depend on the compiled-in libgcrypt [libgcrypt] library. Table 5 list all supported encryption algorithms.

| Name | Description |
|---|---|
| none | no encryption; store as clear-text-data |
| 3DES | 3DES cipher |
| CAST5 | CAST5 cipher |
| BLOWFISH | Blowfish cipher |
| AES128 | AES cipher 128bit |
| AES192 | AES cipher 192bit |
| AES256 | AES cipher 256bit |
| TWOFISH128 | Twofish cipher 128bit |
| TWOFISH256 | Twofish cipher 256bit |

*Table 5: Encryption algorithms*

Usually one of the AES encryption algorithms, e. g. AES 256, is a good choice. For details about the encryption algorithms see the documentation of *libgcrypt* [libgcrypt].

--crypt-asymmetric, -a

Use asymmetric encryption with RSA.

--crypt-password=<password>

Default crypt password phrase to use.

Note: Use this option with care! Password specified on the command line may be stored in the history of your shell, thus can be recovered by displaying the command history.

--crypt-public-key=<file name>

Name of file with the RSA encryption public key.

--crypt-private-key=<file name>

Name of file with the RSA encryption private key.

Note: Because the RSA encryption private key is a critical piece of data, always keep this key at a secret place!

## Splitting

--archive-part-size=<size>, -s=<size>

Create archive parts of the specified size.

Note: The size of a single created part may be slightly bigger than the specified number of bytes. The reason for this are internal buffers used in the compression algorithms which must be stored, too, when an archive part is completed.

## Storage

### *FTP*

--ftp-login-name=<name>

Specify the general FTP login name.

--ftp-password=<password>

Specify the general FTP login password.

Note: Use this option with care! Password specified on the command line may be stored in the history of your shell, thus can be recovered by displaying the command history.

### *SSH*

--ssh-port=<n>

Specify ssh port to use. Default is 22.

--ssh-login-name=<name>

Specify the default ssh login name.

--ssh-password=<password>

Specify the default ssh login password.

Note: Use this option with care! Password specified on the command line may be stored in the history of your shell, thus can be recovered by displaying the command history.

--ssh-public-key=<file name>

Specify the file name with the ssh public key.

--ssh-private-key=<file name>

Specify the file name with the ssh private key.

### *CD/DVD*

--dvd-request-volume-command=<command>

Command to execute when a new CD/DVD is required and should be placed into the CD/DVD drive. Usually this command open some interactive dialog for the user. If a CD/DVD changer unit is available the appropriated command to change the CD/DVD may be send to this unit as well.

--dvd-unload-volume-command=<command>

Command to unload a CD/DVD volume. The default value is the *eject* command.

--dvd-load-volume-command=<command>

Command to load a CD/DVD volume. The default value is the *eject* command.

--dvd-volume-size=<n>[G|M|K]

CD/DVD volume size. Default value are 3.4GB.

Note: Of course DVDs can be completely filled. Nevertheless if *dvdisaster* is used as a post processing tool when creating a DVD the size of 3.4GB is a useful value. The rest of the available DVD space is then used by *dvdisaster* to store Reed-Solomon error-correction codes.

--dvd-image-pre-command=<command>

Pre-precess command to execute before creating a DVD image. This can be some arbitrary command which should always be executed before a new DVD image is created.

--dvd-image-post-command=<command>

Post-precess command to execute after creating a DVD image. This can be some arbitrary command which should always be executed after a new DVD image is created.

--dvd-image-command=<command>

Command to create a DVD image. Default is *mkisofs*.

--dvd-ecc-pre-command=<command>

Command to execute before error-correction codes (ecc) are added to a DVD image.

--dvd-ecc-post-command=<command>

Command to execute after error-correction codes (ecc) are added to a DVD image.

--dvd-ecc-command=<command>

Command to added error-correction codes (ecc) to a DVD image. A useful command is *dvdisaster* to add Reed-Solomon codes to the DVD image.

--dvd-write-pre-command=<command>

Command to execute before DVD image is written.

--dvd-write-post-command=<command>

Command to execute after DVD image is written.

--dvd-write-command=<command>

Command to write a DVD image.

### *Device*

--device=<device name>

Specify device name.

--device-request-volume-command=<command>

Command to request new volume for device.

--device-load-volume-command=<command>

Command to load volume for device.

--device-unload-volume-command=<command>

Command to unload volume from device.

--device-volume-size=<n>[G|M|K]

Device volume size.

--device-image-pre-command=<command>

Pre-precess command to execute before creating a device image. This can be some arbitrary command which should always be executed before a new device image is created.

--device-image-post-command=<command>

Post-precess command to execute after creating a device image. This can be some arbitrary command which should always be executed after a new device image is created.

--device-image-command=<command>

Command to create a device image.

--device-ecc-pre-command=<command>

Command to execute before error-correction codes (ecc) are added to a device image.

--device-ecc-post-command=<command>

Command to execute after error-correction codes (ecc) are added to a device image.

--device-ecc-command=<command>

Command to added error-correction codes (ecc) to a device image.

--device-write-pre-command=<command>

Command to execute before device image is written.

--device-write-post-command=<command>

Command to execute after device image is written.

--device-write-command=<command>

Command to write a device image.

## Server

--daemon

Run BAR as a daemon mode (in server mode).

--no-detach, -D

Do not detach process when running in daemon mode.

--server-port=<n>

Specify server port to use. Default is 38523.

--server-tls-port=<n>

Specify TLS (SSL) server port to use. Default is 38524.

--server-ca-file=<file name>

Specifiy TLS (SSL) server certificate authority file (CA file) to use. Default is */etc/ssl/certs/bar-ca.pem*.

--server-cert-file=<file name>

Specify TLS (SSL) server certificate file to use. Default: */etc/ssl/certs/bar-server-cert.pem*.

--server-key-file=<file name>

Specify TLS (SSL) server key file to use. Default is */etc/ssl/private/bar-server-key.pem*.

--server-password=<password>

Specify server password.

Note: Use this option with care! Password specified on the command line may be stored in the history of your shell, thus can be recovered by displaying the command history.

--server-jobs-directory=<path name>

Specify server jobs directory. Default is */etc/bar/jobs*.

--remote-bar-executable=<file name>

Specify remote executable of BAR binary.

## Misc

--log=<name>[,<name>...]

Specify the information stored in the log file. One or more log information types can specified separated by a comma. Table 6 show the available information log types.

| Log type | Description |
|---|---|
| none | no logging (default) |
| errors | log errors |
| warnings | log warnings |
| ok | log stored/restored files |
| unknown | log unknown files |
| skipped | log skipped files |
| missing | log missing files |
| incomplete | log incomplete files |
| excluded | log excluded files |
| storage | log storage |
| all | log everything |

*Table 6: Log information types*

--log-file=<file name>

Specify the log file name.

--log-post-command=<command>

Command to execute as a post-processing command for a log file. This command can e. g. be a mail command to send the log file via mail.

--pid-file=<file name>

BAR process identifier file to use.

--group, -g

Group files when listing archive content.

--all

Show all files not only the newest version of a file when listing archive content.

--long-format, -L

Use long output format when listing archive content.

--human-format, -H

Print sizes in human readable format (number and unit) instead of bytes.

--no-header-footer

Suppress output of header/footer when listing archive content.

--delete-old-archive-files

Delete all old archive files after creating new archive files.

--skip-unreadable

Skip unreadable files instead of reporting an error.

--overwrite-archive-files

Overwrite existing archive files instead of reporting an error when archive file already exists.

--overwrite-files

Overwrite existing files when restoring files instead of reporting an error.

--wait-first-volume

Wait for first volume (CD/DVD or device). If not specified BAR assume the first CD/DVD or device volume is already available.

--raw-images

Store all block of a disk in an archive instead of used blocks only.

Note: BAR always store all blocks of a disk into the archive. Not used blocks are stored with content "0". When compression is used those blocks almost use no space in the created archive.

--no-storage

Do not store archives (skip storage). This option may be useful to create e. g. incremental file lists only with the option *--create* and *--full*.

--no-bar-on-dvd

Do not store a copy of BAR on CDs/DVDs. By default a copy of the BAR executable is stored on a created CD/DVD, too, to be able to restore the files in the archive easily.

--stop-on-error

Immediately stop when an error occur.

--no-default-config

Do not read personal configuration file *~/.bar/bar.cfg*.

--ecc

Enable generation of error correction codes when creating DVD or device images. For DVDs the external tool *dvdisaster* is required.

--quiet

Be quiet and suppress any output.

--verbose=<n>, -v=<n>

Specify verbosity level. A range of 0..3 can be specified. Default is 1. A higher verbosity level produce more informational output.

--version

Output version of BAR.

--help, -h

Output this help to options.

--xhelp

Output help to extended options.

--help-internal

TODO

## Archive file format

The BAR archive format is organized in so called *chunks*. Each archive consists of a sequences of chunks. All non-data entries are stored in big endian format. The following data types are used:

| | |
|---|---|
| uint8 | unsigned 8 bit integer |
| uint16 | unsigned 16 bit integer |
| uint32 | unsigned 32 bit integer |
| uint64 | unsigned 64 bit integer |
| int8 | signed 8 bit integer |
| int16 | signed 16 bit integer |
| int32 | signed 32 bit integer |
| int64 | signed 64 bit integer |
| string | string: uint16 length field, following string character data aligned to a 2-byte boundary |
| crc32 | unsigned 32 bit containing the CRC32 sum of the entry |
| data | arbitrary data |

A single chunk have the structure:

- ID (4 characters)
- size (uint64)
- data (n bytes)

The chunk ID is unique and specify how the data of the chunk must be interpreted. Size give the size of the chunk in bytes excluding the header (id and size). Chunks can contain sub-chunks which have the same structure.

Note: The file *bar/archive_format.def* contain the formal specification of all archive chunk types. This specification is compiled in to C source code with the perl-script *bar/archive_format.pl*.

## Compilation

To compile BAR by yourself you need a gcc compilation environment, make and some libraries. The following list the required packages to compile BAR including all features. If some package is not available, some features may be disabled. If a mandatory package is missing *configure* will

output an error:

- gcc
- openjdk-6-jdk[2]
- make
- epm
- zlib1g
- zlib1g-dev
- libbz2-1.0
- libbz2-dev
- liblzma
- liblzma-dev
- libssl0.9.8
- libssl-dev
- libssh2-1
- libssh2-1-dev
- libgcrypt11
- libgcrypt11-dev
- gnutls-bin
- libgnutls26
- libgnutls-dev
- ftplib3
- ftplib-dev

To compile BAR usually the command sequence:

*./configure*
*make*
*make install*

will create and install a BAR binary on your system.

# Frequently asked questions

If you try to run or recompile BAR by yourself and you run into some problem, please check the following question and answers list before creating a bug report. Some problems are known and are not caused by BAR.

1. *Why is BAR linked static by default?*

   BAR is not fully linked static, but partially. All needed libraries are linked static except *libc*, *libm* and *libpthread*. The advantage of this type of linkage is that BAR can be executed on many systems, even when some specific libraries like libraries for compression or encryption are not available on the target system. BAR can also be started from a CD or DVD without installation. If a dynamic linked BAR version is required set the configure option *--enable-link-dynamic*.

2. *Can I build a dynamic linked version of BAR?*

   Yes. Use *--enable-link-dynamic* when calling configure. The resulting binary is linked

2 Instead of the OpenJDK package a standard JDK package may be usable, too.

dynamic.

3. *Can I create incremental archives even the file system does not have an "archive" bit?*

   Yes. BAR create his own incremental list files (*.bid* files). These list files contain information which are used to check which files changed.

4. *Can I just create the incremental list file?*

   Yes. Use option *--no-storage*. The incremental list file is created, but no archive files are neither created nor stored.

5. *I cannot establish a TLS/SSL connection with barcontrol.*

   For the TLS/SSL connection in barcontrol a TLS/SSL key is required. Create that key once with "make keys" and copy the keys with "make install_keys" in the appropriated directories of your system or use the command *bar-keygen*.

6. *I tried to list a *.bid-file, but it does not show anything.*

   The *.bid*-files are not archive files, instead these are files with lists of file names needed for ab incremental backup. Do not modified, delete or copy those files. The content is not encrypted. BAR does not store them on an external server or CD/DVD.

7. *When I specify -# /foo/ the list of included files is different to -# /foo. Why?*

   The option values for -# and *-!* are patterns, not file or directory names.*-# /foo/* match to everything which starts with */foo/* including the trailing "/". If the directory */foo* contain sub-directories only those sub-directories are included. If you specify *-# /foo* the directory foo including all sub-directories and all files in "/" starting with "foo" are included.

8. *I get the error "configure: error: function zlibVersion() is not available".*

   The zlib library [zlib] is mandatory. Please install the zlib package (binary and development files) with your distribution package tools.

9. *Instead of the SWT JAR included in BAR I like to use another one I already have. BARControl is starting, but some of the buttons does not work properly. What is wrong?*

   This is probably a bug in the SWT JAR you are using in combination with the installed GTK libraries on your system. Probably SWT 3.5.x and GTK 2.18 or newer versions seems not to work together anymore. I could avoid this problem by using SWT 3.6.

10. *When I compile BAR with the libgcrypt library installed my system, the private key file cannot be read. I get the "Unable to initialize private key file from file". Whats wrong?*

    I also saw this problem and I assume it is a bug in some libgcrypt versions. When I use libssh2 with the compiled-in libgcrypt-functions and OpenSSL (configure options --witout-libgcrypt and --with-openssl) this error disappears.

11. *When I compile a static linked version of BAR, I get linker errors for some libraries. What is wrong?*

    Maybe there is no static version of some external library available on your system. Try to download the required libraries with the script *download-third-party-packages.sh*. These libraries are linked static.

12. *When I create the key files with bar-keygen it takes a very long time. Why is this so slow?*

    If some specific version of *certtool* is used to generate the key files it may use */dev/random* instead of */dev/urandom* to get random data for generating the keys. Reading */dev/random* may block when there is not enough entropy to generate new random numbers. This slow down the key generation. Try to install OpenSSL and use the option *--openssl* when calling *bar-keygen*.

# Contact and bug reports

If you find a bug in BAR, please send me a bug report including the following information:

- version number of BAR
- description of your system environment (Linux distribution and version)
- if you used a pre-compiled version or source-version
- steps to trigger the bug
- what is going wrong

In case of a fatal error where BAR is crasing, please send me – whenever possible – a C stacktrace or a Java stacktrace. To get a C stracktrace you must compile BAR with the configure option --en*able-debug* and start BAR inside a debugger like *gdb*. You get he C stacktrace after the crash with the *gdb* command *bt*. For a Java stracktrace execute BARControl with the option *--debug*.

Send the report to:

torsten.rupp@gmx.net

I will try to reproduce the problem and fix it as soon as possible.

# License

BAR and all included files are under the GPL version 2. The full GPL version 2 license text can be found here:

http://www.gnu.org/licenses/gpl-2.0.html

barcontrol is using SWT [SWT]. SWT is under the "Eclipse Public License" which can be found here:

http://www.eclipse.org/org/documents/epl-v10.php

# References

| | |
|---|---|
| ant | http://ant.apache.org/manual |
| bzip2 | http://www.bzip.org |
| cdrtools | http://cdrecord.berlios.de/old/private/mkisofs.html |
| dvd+rw-tools | http://fy.chalmers.se/~appro/linux/DVD+RW/ |
| dvdisaster | http://dvdisaster.net/en/index.html |
| epm | http://www.epmhome.org/ |
| ftplib | http://nbpfaus.net/~pfau/ftplib |
| gnutls | http://www.gnu.org/software/gnutls |
| growisofs | http://fy.chalmers.se/~appro/linux/DVD+RW |
| JRE, JDK | http://www.java.com |
| launch4j | http://launch4j.sourceforge.net |
| libgcrypt | http://directory.fsf.org/project/libgcrypt |
| libssh2 | http://www.libssh2.org |
| LZMA | http://tukaani.org/lzma |
| OpenSSL | http://www.openssl.org |
| SWT | http://www.eclipse.org/swt |
| zlib | http://www.zlib.net |

# Alphabetical Index