

*Document revision: 2009 09 09*

**Author:** Giorgio Tani

**Translation:** Giorgio Tani

**This document refers to:**  
PeaZip 2.7 executable implementation;

**Licensing:**  
present documentation is released under GNU GFDL License;  
PeaZip executable implementation is released under GNU LGPL License.

**PeaZip official site:**  
<http://sourceforge.net/projects/peazip/>

**For more information about the licenses:**  
GNU GFDL License, see <http://www.gnu.org/licenses/fdl.txt>  
GNU LGPL License, see <http://www.gnu.org/licenses/lgpl.txt>



## Content:

• How to...	3	Basic
• What is PeaZip	4	
• File manager	5	Advanced
○ Enter password	13	
○ Set advanced filters	13	
○ Create keyfile	14	
○ File tools	15	
• Extract archives	16	
• Create archives	18	
• PeaLauncher	23	Expert
• Settings	24	
• Supported formats	27	Additional information
• Customisation and scripting	29	
• Translations	30	
• Notes	31	

# How to...

This mini-tutorial introduces the most common operations that can be performed through PeaZip, following chapters contains a more detailed explanation of the application and of terms used here.

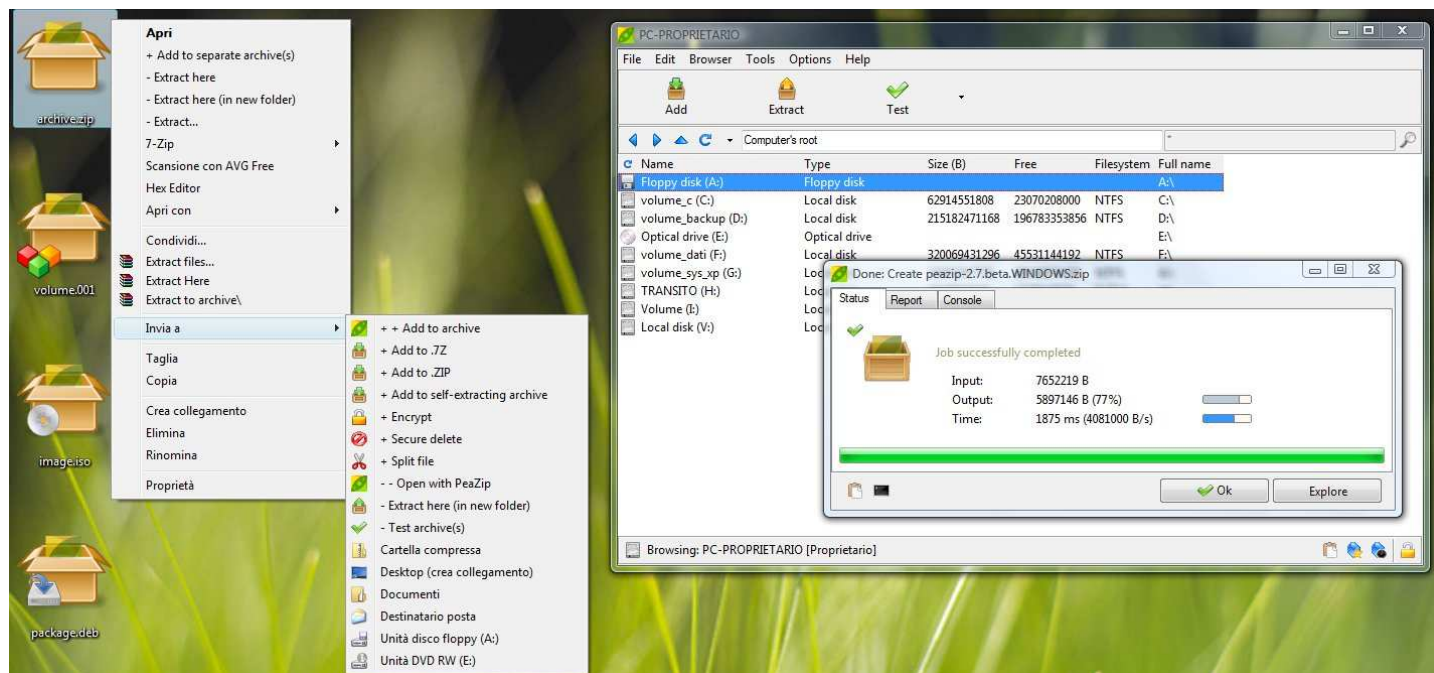


Image 1: high resolution icons, SendTo and context menu integration (PeaZip entries conventionally starts with + and -) and some program's windows in Windows Vista

## CHANGE LANGUAGE

- User interface is available in different languages: Options > Localization (and in Options > Settings)

## EXTRACT ARCHIVES

- From the system: rightclick on the archives and click "Extract here" or "Extract here (in new folder)" from context menu, or "Extract..." for more options (extraction path, password...)
- From PeaZip (recommended): select archives and click on "Extract" button, or rightclick and click "Extract" or "Extract (in new folder)" in context menu

## EXTRACT OBJECTS FROM ARCHIVE

- Doubleclick the archive to open it with PeaZip, select files and folders to be extracted and drag them to the system, or rightclick and chose "Extract selected" in context menu

## CREATE ARCHIVE

- From the system: rightclick on objects to be archived and click on "Add to archive" in SendTo menu
- From PeaZip (recommended): select objects to be archived and click on "Add" button

## CREATE SEPARATE ARCHIVES

- From the system: rightclick on objects to be archived separately and click on "Add to separate archives" in context menu
- From PeaZip (recommended): select objects to be archived separately and click on "Add" button; before confirming with "Ok", check "Add each object to a separate archive" option

## UPDATE EXISTING ARCHIVE

- Doubleclick to the archive to open it with PeaZip and drag here files and folders to be added, or click on "Add" button and use application's context menu to add objects to the archive

# What is PeaZip

PeaZip is a general purpose file and archive manager application, aiming to provide a cross-platform graphical interface for many Open Source archiving and compression utilities in order to handle most of available archiving formats like 7Z, RAR (extraction), TAR, ZIP and many other ones, see **Supported formats** chapter for more information.

The program features powerful and flexible inclusion/exclusion filters and search tools, provide optional two factor authentication through password and keyfile, and allows to deeply fine tune the job's definitions, exposing through a single, consistent frontend GUI the options of underlying applications.

The list of objects to be archived or extracted can be saved for future use, to speed up backup and restore tasks. Also the resulting command for archive creation and extraction can be saved, to get the full control on job's definition, helping the user in bridging the gap between GUI and console applications to get the best of both worlds. A detailed log is available after each operation.

PeaZip also collects a set of handy file management tools: robust file copy, split and join files, fast or secure file deletion, calculation of a wide set of checksums and hashes over selected files, byte-to-byte comparison of two files, web search etc.

PeaZip can be used as file manager, or can be used from **context** and **SendTo menu**.

File associations and menu entries (both for context and SendTo menu) can be changed running the setup program any time it is needed.

*Hint: on Windows systems you can run the installation as administrator with runas command, or "Run as" entry in system's context menu; on Windows Vista UAC will automatically ask for running the process as administrator.*

If no system integration is preferred, **PeaZip Portable** is available as standalone application, not needing installation and not modifying the host system, both packages are available on application's website main page:

<http://peazip.sourceforge.net/>

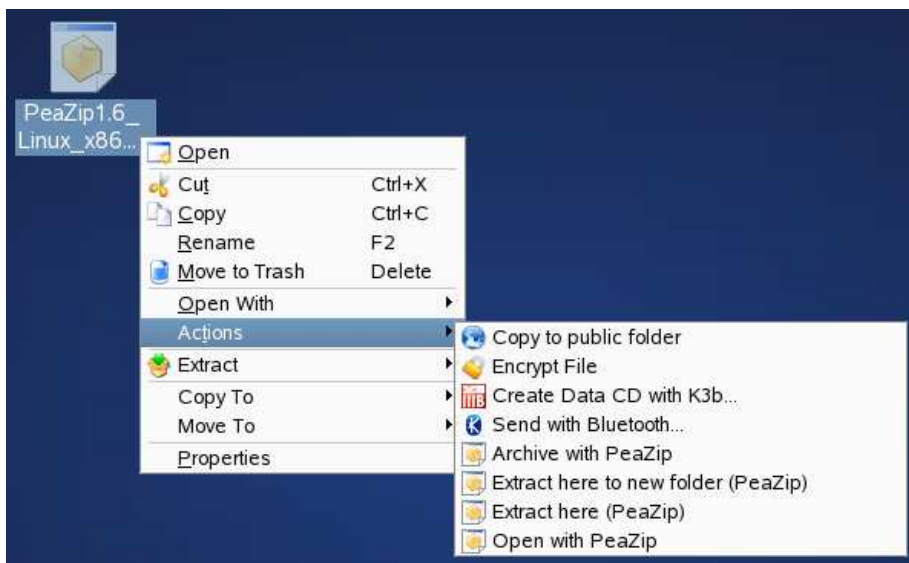


Image 2: menu integration in Linux (KDE, Konqueror Service Menus)

All the open source backend applications included in base packages contain only open source software released under OSI-approved licenses.

Handling formats not supported through open source software (presently, only ACE archives) requires installing separately available plugin, which contains closed source, royalty free binaries.

The list of available plugin can be browsed alongside the release map of PeaZip.

# File manager

The application starts by default with **file manager** interface, pointing to the last visited directory, for navigation in the filesystem and in archives.

*On start-up, PeaZip parses the input parameters trying to understand to what function they should be passed to (i.e. to open an archive for browsing, or adding selected objects to a new archive), see “**Customisation and scripting**” chapter for the startup parameters that can be passed to PeaZip, if you want to use it in scripts or customize system’s integration (registry entries, SendTo menu links etc...)*

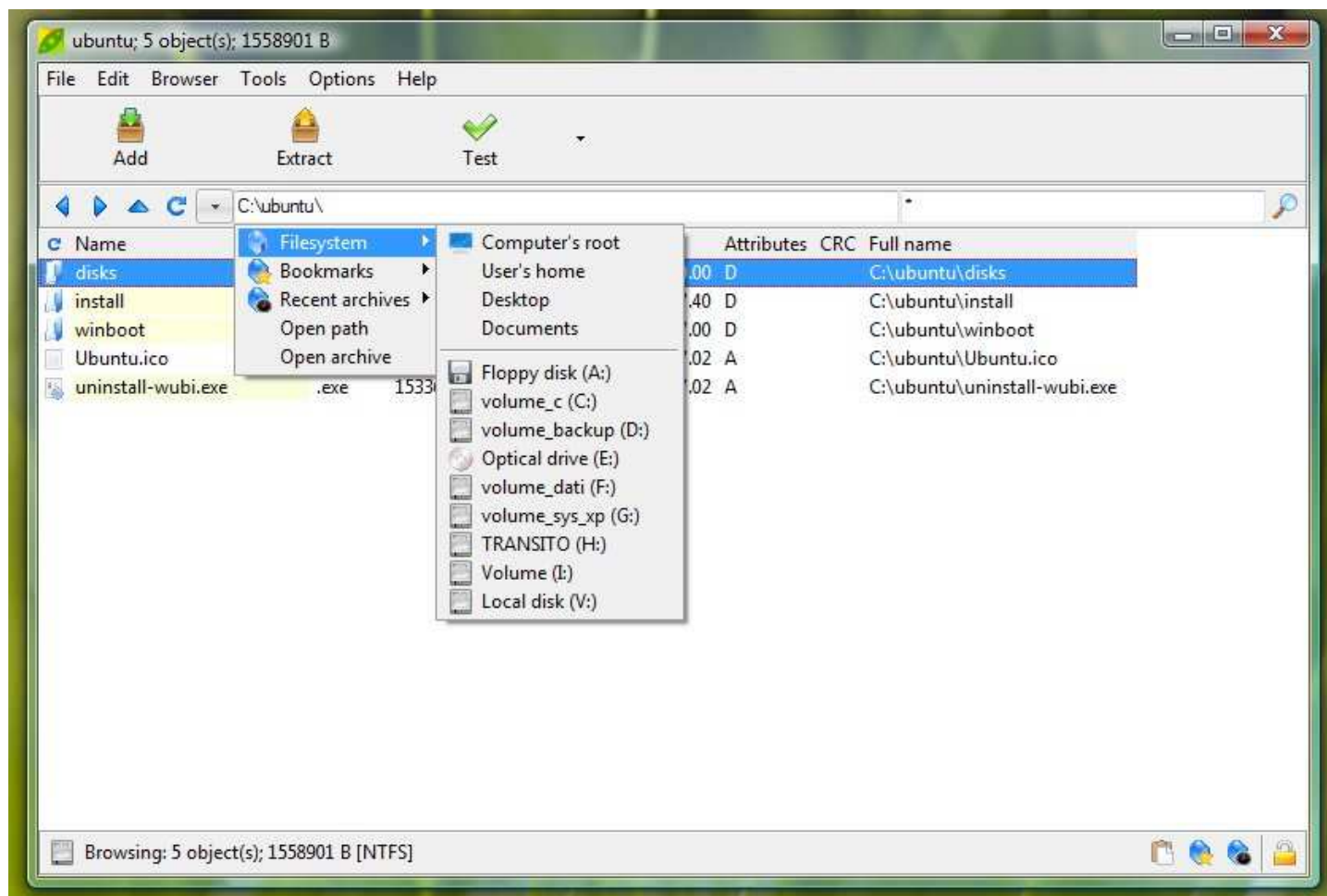


Image 3: browser showing navigation menu

Application's main menu features:

**File** submenu, contains primary application's functions.

In the first area, **Create archive** activates the archive creation interface, which allows to add files and folders to the archive's layout and to save, restore and merge layouts for further use.

In the second area are featured Open path and Open archive entries, and **Filesystem**, **Bookmarks** and **Recent** submenus which represent three alternative ways to quickly access to most used archives or folders in the browser interface.

- **Filesystem** menu is organized following a functional and hierarchical point of view, featuring links to commonly used paths like root, home, desktop, and documents, links to "Open path" and "Open archive", and links to mounted devices.
- **Bookmarks** menu reflects user's point of view, storing user defined favourite files, folders and search definitions; last entry activates Bookmarks panel allowing to add, edit, sort and remove bookmarks. Items can be added to bookmarks also from file browser's context menu "Bookmarks" entry, from History panel's context menu, and from layout composer interface ("Misc" entry in context menu).
- **Recent archives** menu is the chronological point of view, storing last accessed archives (this feature can be disabled for privacy, see "Settings" chapter); last entry activates History panel which contains current session's visited paths, archives and search definitions.

**Reduce to tray** entry send PeaZip to tray area; right clicking on the tray icon it is possible to resume the program, or access most common program's functions.

**Edit** submenu changes the selection of currently displayed files, for date, size, extension, attributes etc...

**Browser** submenu, featured only when file manager is displayed, contains

- jump to archiving and extraction layout interfaces
- **refresh** (F5) forcing refresh of currently displayed content
- **toggle browser/flat view** (F6), flat view displays all together the objects contained in the current path or in the archive

**“Tools”** submenu contains:

- **Enter password / Keyfile** (F9) sets the default password (and optionally keyfile, if two factor authentication is desired) to be used in browsing, testing, extraction and archive creation
- **Set advanced filters** (F11) sets multiple inclusion and exclusion filters to be used in browsing, testing, extraction and archive creation; filters are applied only to archive formats managed through 7z backend interface, see “Supported file types” chapter
- **Create keyfile** (F12) allows to sample entropy from the system and from user’s actions to generate a random keyfile; this utility can be also used to generate random passwords to be used in any other application/website/etc
- **System tools** submenu, collecting system’s disk utilities (clean, defrag, manage, remove), system management tools (control panel, computer management, task manager) and display environment variables (both for Linux and Windows).
- **System benchmark** utility to rate the host system in terms of MIPS (millions of integer instructions per second) and Core 2 Duo equivalent speed in MHz

**“Options”** submenu contains:

- **Run as different user** entry (Windows only), which closes current PeaZip instance and opens a new one with alternative user profile;
- **Localization**, to quickly change applications language;
- **Settings**, to customize application’s behaviour.

**“Help”** submenu points to project’s website and to most up to date documentation available online, as well as offline documentation included in the package: PeaZip help (linked also to F1 functional key), and a short tutorial introducing most common operations.

The **toolbar** features:

- **“Add”**
  - While browsing the filesystem, the button adds selected files and folders to the current archive layout; before confirming the creation of the archive with “Ok” it is possible to modify the list of objects to be archived (dragging them or using the context menu) as well as other options, output name etc, see “Create archive” chapter for more information.
  - While browsing an existing (writeable) archive, the button brings to the archive update interface; it is possible to add files and folders to be added to the archive as in the previous case, dragging them or using context menu.
- **“Extract”**
  - Extracts all selected archives at once (or the current archive being browsed); before conforming the extraction with “Ok” it is possible to modify output path and other options
- **“Test”**
  - Test selected archives for integrity

On the left of the “Test” button, an arrow shows a menu with other functions, explained in details in context menu section of this chapter:

- List and Info to display information (more detailed in the second case) about the content of selected archives, files or folders.
- file manager functions (only while browsing files): Copy to and Move to, Quick delete and Secure delete, or
- Delete from archive (only while browsing archives)
- Explore path and Open command prompt here, to open the path being currently browsed with Explorer (or other default file manager) and command prompt respectively



Image 4: toolbar, note, on the right of the last button, the dropdown button which shows additional functions

To quickly jump to desired directory or archive PeaZip offers a **navigation menu** (shown in Image 3) in the navigation bar (image 5); the menu is shown clicking on the arrow on the left of address.

The navigation menu is organized in three sections (as in main menu), and can be reached also from application’s context menu and status bar.



In the browser's **navigation bar** the blue arrows (back, forward, and up) navigate in previously visited path (or any previously applied search filter), as displayed on "History" panel, and to go to upper level. Right clicking on one of the 3 arrows (back, forward and up) displays a popup menu to quickly jump to one of the 8 more recently visited paths/filters.

Clicking "Refresh" icon (or menu entries, or F5), forces the browser to update; while the browser is updating the refresh icons show an animation, and the refresh icon is greyed until the browser is ready.

The magnifier icon in navigation bar can be used for basic recursive and non recursive **search** functions; \* (string) and ? (single character) wildcards are allowed.

This basic search filter is overridden by advanced filters (F11) only for archive types allowing inclusion/exclusion filters (i.e. 7z, rar, tar, zip...).

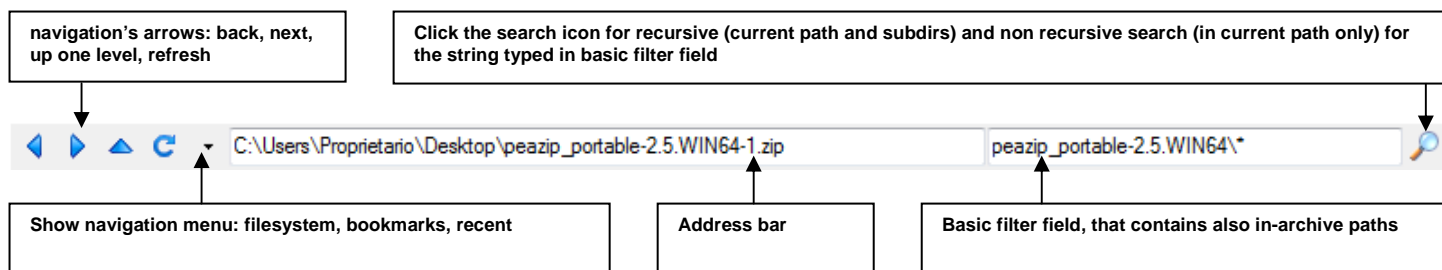


Image 5: navigation bar (Firecrystal theme)

At the bottom of the application's window, on the left of the **status bar**, an icon represents the current path (or archive): the icon's hint will give information about the current unit (or archive), clicking the icon will give more detailed information, and rightclicking the icon will show the navigation menu. On the right of the icon it is displayed synthetic information about the currently displayed content.

On the right of the status bar are provided some more functional buttons:

"**Clipboard**" shows cut/copied items and clipboard's options, in the bottom part of the file browser.

Clipboard content's table allows to check objects currently cut and/or copied, and to remove single objects from clipboard if it's need to refine the selection.

Clipboard behaviour can be switched between two modes:

- **Standard clipboard** (default) behaves like usual file browser's clipboard, allowing a single cut or copy operation. Any further selection replaces the previous one, and on paste operation cut objects, have been moved, are removed from clipboard, while copied objects are kept in clipboard.
- **Advanced clipboard** allows to store multiple (and mixed) cut and copy operations; any selection is added to the previous ones (if objects are duplicate, previously selected are kept), even from different paths and disks, and executed on paste operation, which clears the clipboard content.

"**Bookmarks**" shows bookmarked files, folders and archives.

Bookmarks can be dragged to arrange them in the desired order, clicking on the leftmost column of the Bookmarks table; otherwise the bookmarks can be arranged alphabetically by bookmark's path or by description clicking on the respective column's header (clicking again will invert the order).

"**History**" shows current session's history panel.

The splitter between this area and the file browser can be used to adjust the relative sizes of the two areas, and clicking on the status bar will hide bottom panel.

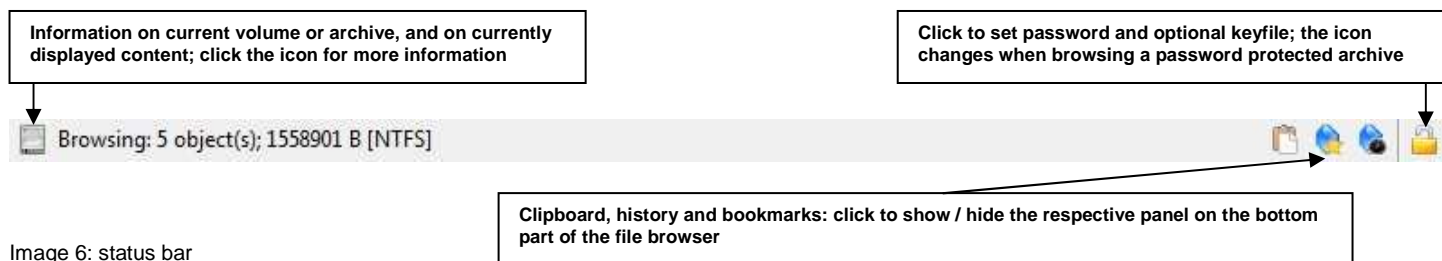


Image 6: status bar

"**Enter password / keyfile**" (F9) allows to set password and keyfile to be used when operation on archives, browsing, extracting/testing or to create encrypted archives; this information is kept for the current session, and can be different for each separate instance of the application.

Encrypted content, while browsing an archive, is shown with a '\*' appended to filename.

If the archive is encrypted and the password is not provided, a popup will ask the user to enter the password/keyfile when extract/test/list operation are attempted.

If the password/keyfile is set, the locker icon is changed to highlight that.

Keyfile is not mandatory, it can be used if two factor authentication is preferred to password-only authentication; keyfile creation utility can be launched from Tools menu or with F12 functional key.

If the directory structure area is encrypted (i.e. .7z archives created with `-mhe` option), browsing is not possible until the correct password/keyfile is provided: the archive browser will be empty and “no matches” will be displayed in the status bar until it becomes possible to browse the archive, having the user provided the right password.

Please note the same can happen when the archive cannot be browsed for other reasons, i.e. it was corrupted due to bad download or storage media failure. When strong encryption is involved, it may not be possible to determinate if the the provided password is incorrect or if the file is corrupted, since resulting output will be random-looking in both cases.

The file browser supports **drag and drop from system to application**.

When files and folders are dragged to file manager, they get listed in the archive creation interface (as if they were selected and added with “Add” button), allowing to fine tune the job before confirming, or cancel the job.

In the same way, objects dragged into the browser while browsing an archive will be added to the current archive, if the file type allows modifications. In example, it will not be possible adding objects to archive types supported only for reading or to some solid archives.

If a single archive file is dragged to the browser PeaZip will show a disambiguation popup to ask if adding the object to the archive or rather opening it.

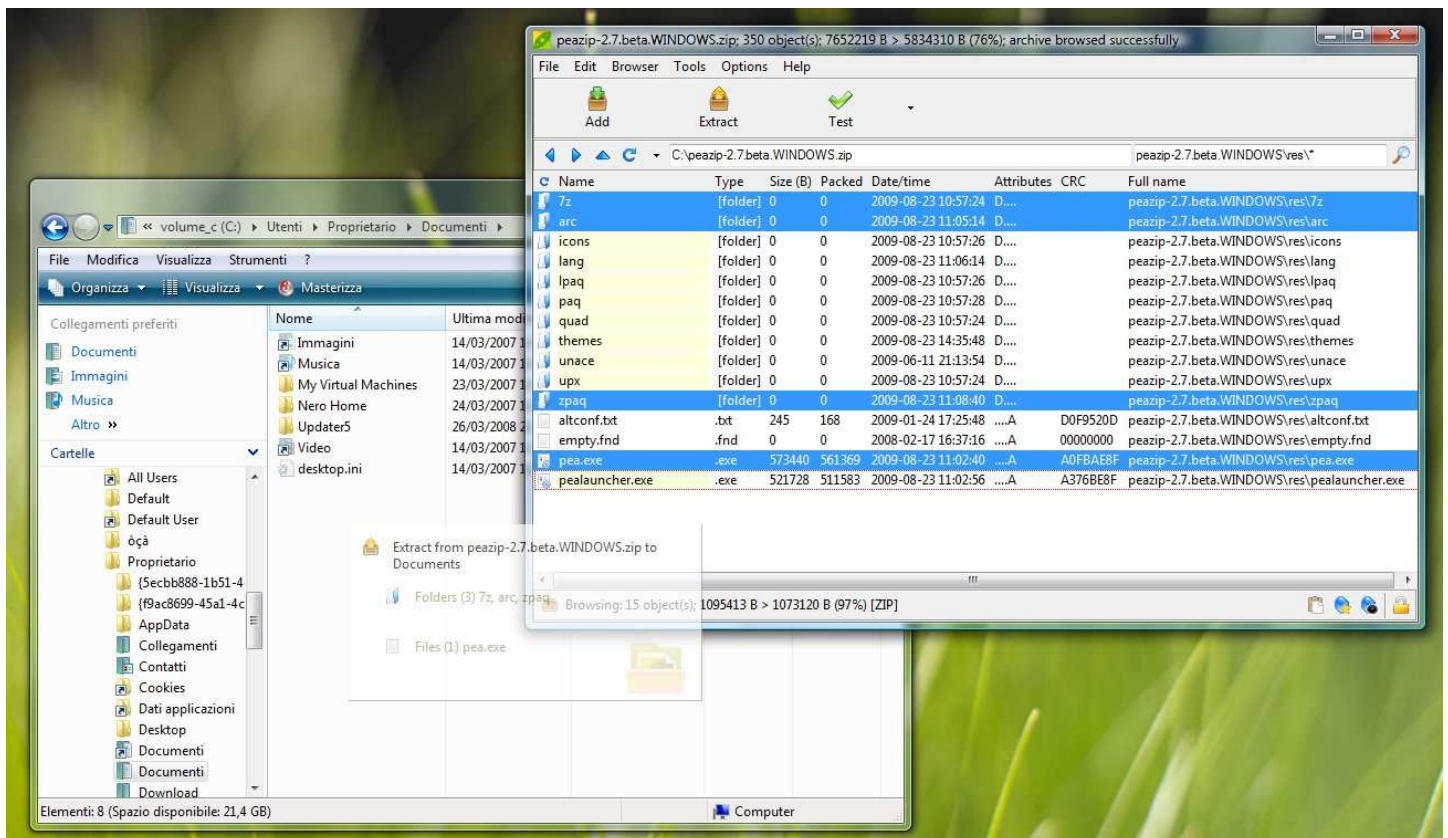


Image 7: drag and drop from application to system (Windows only); files and folders can be dragged from the filesystem or from archives to the desktop or (file)Explorer's windows with Address field enabled (as it is by default on all Windows versions). Drag and drop operations, as well cut/copy selections, are cancelled pressing Esc

Browser interface features also a custom **drag and drop to the system** function (in Windows only, Image 7).

Dragging an archived file or folder from PeaZip to the system will extract it to the desired location; if the file is not contained into an archive it is copied to the desired location instead.

The custom drag and drop to system feature doesn't need to copy files being dragged to system's temp folder before, resulting in faster operation when big files are involved, and in better security if temp folder has not the same desired security policies of actual output folder.

This drag and drop implementation will not show default Windows drag and drop cursors, but instead it will show a transparent information area following the mouse and reporting detailed information on content being dragged, and it can drag files to the path of (file)Explorer windows with Address field enabled (as it is by default on all Windows versions), or to the desktop; it will prompt a directory selection dialog if the path is not recognized i.e. content is dropped to an application other than (file)Explorer.

In the file browser, clicking on **titles bar** column's header sorts displayed objects by the selected column i.e. name, full (file and path) name, extension, date, size etc; a second click inverts the order.

Archives and folders are highlighted (respectively, green and orange), such objects can be opened for browsing with double-click.



Archives contained into another archive will be opened in a separate instance of PeaZip; please note that by default this is a “preview” operation, extracting data to a temporary folder (by default in the same path, if writeable).

*Hint: to open multi volume archives, the first volume (i.e. the one with .001 extension) must be selected as input file.*

*Hint: unsupported file types can be forcedly open as archives in PeaZip as custom format: this allows to use an arbitrary binary, and to customize the command's syntax, for dealing with them.*

The browser can switch between **flat view** mode, displaying all objects contained in the archive, and classic browser mode, using “Toggles browse/flat view” (F6) in main and context menu.

Flat view is used also when performing basic search or applying advanced filters; opening a folder will browse it in classic mode, exiting the flat view mode.

*Hint: some applications don't explicitly declare the name of directories contained in the archive; in this way PeaZip cannot list those undeclared objects. Switching to flat mode, or using search or filter feature, will allow seeing all the content of those malformed archives. Extraction, listing and testing of the archive is not affected by this issue.*

*ACE, ARC, PAQ/LPAQ, PEA and QUAD/BALZ archives can be browsed by PeaZip in flat mode only.*

In “Options > Settings > Open archive” it is possible to set PeaZip to start browsing (archives) as classic browser, as flat view or to remember last used mode. Browsing of the filesystem, rather than of archives, is not affected by this setting and will always start in classic mode since listing a path in flat mode could take very long time (i.e. if it is a root folder), and the user will be warned of that when switching to flat mode is requested while browsing the filesystem.

The browser's **context menu**, activated right clicking on the archive browser area, is context sensitive and provides different options while browsing the filesystem and archives of various types (which supports different operations).

In the top area it is featured “**Add to archive**” entry: while browsing the filesystem it will send selected files and folders to the archive layout, while browsing an archive it open a submenu allowing to add files or folders or to open a search dialog from which files/folders can be dragged into the archive itself.

In this case objects will be archived starting from archive's root, compressed and encrypted accordingly to the archive's settings.

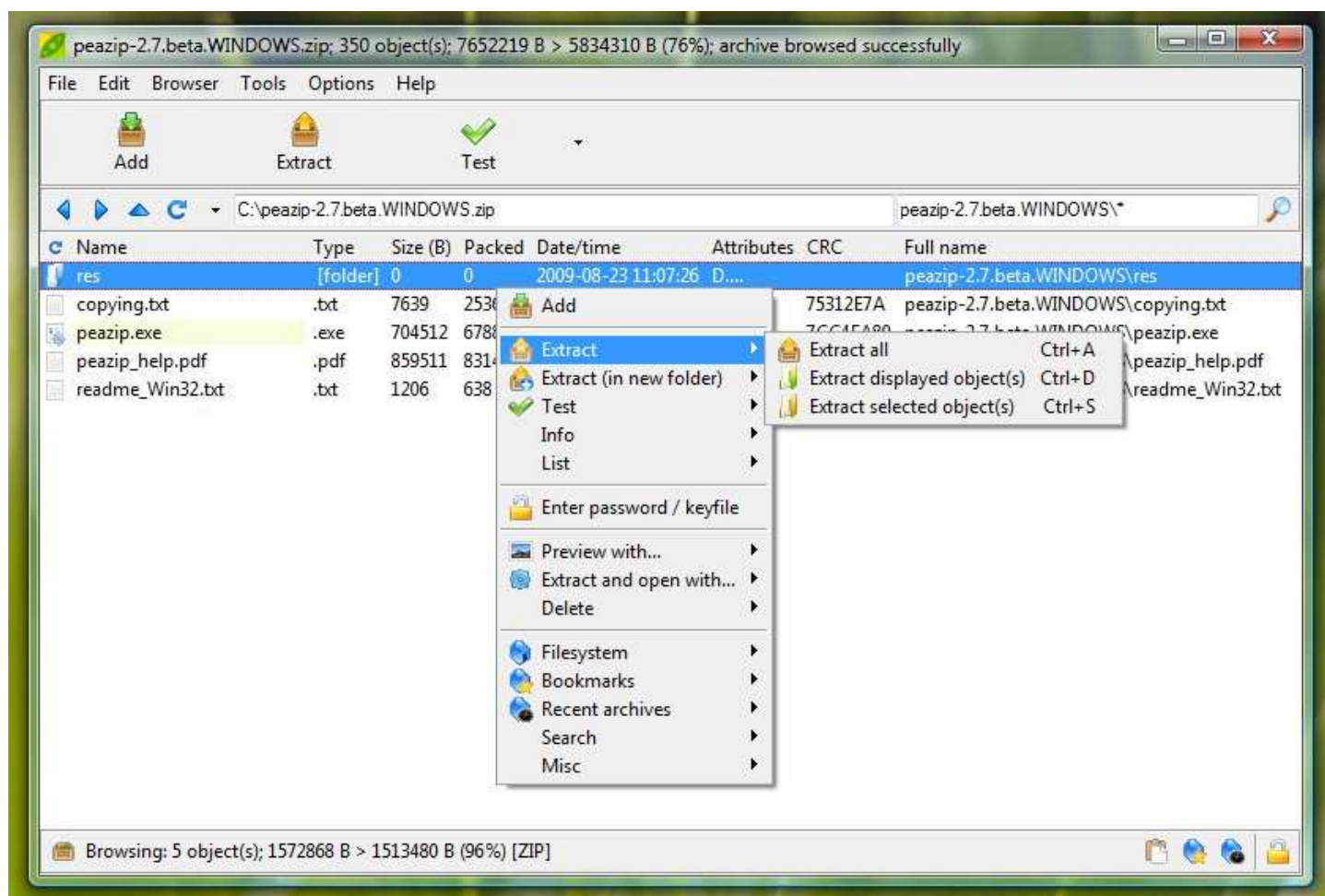


Image 8: the browser, showing context menu. Click on the menu entry or on the locker icon to set password (and optionally a keyfile) to be used to extract or add files to the archive. Some archive types needs password to be set before showing the content of the archive for browsing.

*Note: currently adding objects to archive adds them in archives root.*

*If it is needed to add objects to a subfolder of the archive, a workaround is to place the files in a folder (or nested folders) with that name, and then add that folder to the archive: this will place the file in archive's subfolder with same name. I.e. to add thisfile.txt to \thisfolder1\thisfolder2\ subfolder in the archive, create thisfolder1 (where you prefer on the disk), create thisfolder2 into thisfolder1 and move (or copy) thisfile.txt into thisfolder2. Adding or dragging thisfolder1 to the archive will place thisfile.txt in the desired subfolder \thisfolder1\thisfolder2\ of the archive.*

Please note that if it is possible for the archive format to store objects with different passwords into the same archive (i.e. in .7z format), you can set the different passwords each time you add objects.

*Note **adding files/folders to encrypted .7z archives**: .7z archive format can store objects encrypted with different passwords in the same archive, so when adding an object to an encrypted 7z archive the object will be encrypted with the password/keyfile currently set (in the form which popup clicking on the locker icon). If the .7z archive is encrypted with "Content and filenames" option, objects can be added only using the same password / keyfile for the whole archive.*

By default, if same object exists in the archive (same name and same path in archive's directory tree), the object will be updated: if the object the user is adding is newer, it will replace the older object; if it's older, the archive will not be updated.

Please note that for some archive types, notably most solid archives (i.e. .7z created with solid archive switch enabled), or archive types supported only for browsing extraction (i.e. CAB, RAR...), doesn't allow neither add/update nor delete operations.

The second area provides **archive management related features**: extract, extract in new folder, test, list and info (verbose listing of properties of archived objects).

While browsing archives those entries are transformed in submenu allowing applying the action to the whole archive, or only to displayed or selected objects.

*It's recommended to use "displayed" selection to operate on all displayed objects rather than extend the selection to all objects, because it's faster for the user and because it allows to compose the command in a more elegant way, using current filter definition rather than enumerating all objects.*

The third area contains **file-related functions**: "Open with" group allows to chose if opening the selected file or folder with a new instance of PeaZip, or with associated application or with custom application, or with up to 16 user-selected custom applications and scripts.

*Hint: in Tools > Settings it is possible to define two sets of up to 8 custom application or scripts each (i.e. editors, players, antivirus/antimalware etc) to be used to open files and folders bypassing the default system's file associations; by default PeaZip tries to find some of most common applications to valorise the entries.*

While browsing an archive, "**Open with**" submenu is replaced by "**Extract and open with...**" and "**Preview with...**" submenus; preview functions will extract selected objects to a temporary folder rather than to the output path, and then take the programmed action on the output.

*Hint: if the archive is in a read only path, preview functions will transparently switch to user's temporary folder, while extract here functions will warn the user and ask to select a writeable output path.*

Double-clicking an object while browsing the filesystem is triggers open with associated application action, while browsing an archive it triggers preview with associated application action.

When performing any of those actions while browsing archives, the selected object will be extracted without replicating the directory structure in the archive; if replicating the archive structure is desired uncheck "Always ignore paths for Extract and..." option (see "Settings" chapter). If the object is a directory paths will not be ignored, this condition overrides all other switches.

In this third area, "File tools" submenu contains a set of file management tools described in its own chapter.

"Modify" submenu contains: create new folder, rename, copy or move to, cut, copy, paste (using robust file copy/move with system's **robocopy**, or xcopy on pre-Vista systems).

Cut, copy and paste operations can also be performed with keyboard shortcuts of Ctrl+X, Ctrl+C and Ctrl+V respectively. Pressing Ctrl+V while browsing an archive adds objects from clipboard to the archive; in this case objects will not be removed from the filesystem even if Ctrl+X was used, since PeaZip tries to not automatize potentially dangerous operations such file deletion, letting that kind of decision under the full control of the user.

"Delete" submenu contains quick delete, which allows the **fast deletion** of selected objects, without needing to move them to recycle bin, and secure delete (only in the filesystem, not available in archives) which performs a secure file deletion as described in "File tools" chapter.

While browsing archives filesystem related functions (including cut, copy, paste) are disabled and the area shows only simple deletion from archive, if this action is supported for the current archive type.

Fourth area contains **browsing-related entries**, replicating the navigation menu structure previously described (filesystem/bookmarks/recent).

“Search” allows recursive (search in subpaths too) and non-recursive search from current path, and features “Search on the web” submenu which allows searching for the filename (editable before launching the search) on different web based services, as Google and Yahoo search engines, Usenet, Wikipedia, Wiktionary and other Wiki projects.

This feature can help users in case of any doubt or need of any additional information about the object before archiving it or before extracting it from the archive..

“Misc” contains entries to open a system’s explorer window or a system’s console session in current path.

## File manager’s keyboard shortcuts

File/archive browser supports following keyboard shortcuts; some functions are format-specific and will be ignored if not supported for the current archive type.

### Functional keys:

F1	help
F2	browse desktop / Ctrl+F2 browse user’s home / Shift+F2 browse computer’s root / Ctrl+Shift+F2 browse archive root, if browsing inside an archive (otherwise browse computer’s root)
F3	recursive search / Ctrl+F3 non recursive search (search here)
F4	up one level
F5	refresh
F6	toggle browse/flat view
F7	browse most recently visited item (Ctrl, second, Shift, third)
F8	browse first item in bookmarks list (Ctrl, second, Shift, third)
F9	set password/keyfile
F10	menu
F11	set advanced filters
F12	create keyfile or random password

~	!	@	#	\$	%	^	&	*	(	)	-	+	Backspace
Tab	Q	W	E	R	T	Y	U	I	O	P	{	}	
Caps Lock	A	S	D	F	G	H	J	K	L	:	"	Enter	
Shift	Z	X	C	V	B	N	M	<	>	?	Shift		
Ctrl	Win Key	Alt									Alt	Win Key	Menu
													Ctrl

### Navigation:

Toggle browse mode / flat view mode  
Go to computer’s or archive’s root  
Search (in this folder and subfolders)  
Search in this folder only  
Browse most recently visited item  
Browse first item in bookmarks list  
Open directory/archive  
Up one level  
Go to object’s path  
Go back in history  
Forward in history

\* or F6  
Ctrl+R  
F3  
Ctrl+F3  
F7  
F8  
< or Enter or doubleclick on the folder/archive  
> or Ctrl+U or backspace or click on blue arrow icon or F4  
Ctrl+P (useful in flat view and search/filter mode)  
Ctrl+B or Backspace  
Ctrl+F

### Extract:

Extract all content  
Extract displayed content  
Extract selected content  
Extract to new folder functions  
Extract selected  
Extract selected to new folder

Ctrl+A  
Ctrl+D  
Ctrl+S  
same as previous ones, using Ctrl+Alt+A/D/S  
Ctrl+Enter  
Shift+Enter

Test all (when browsing an archive) Ctrl+T

*Note: "extract selected" extracts the entire selected archive(s) if browsing the filesystem, and extracts selected item(s) if browsing an archive.*

#### Extract and open / preview (on selected objects):

Extract and open with PeaZip	Ctrl+Z
Extract and open with default application	Ctrl+O
Extract and open with ...	Ctrl+W
Preview functions	same as previous ones, using Ctrl+Alt+Z/O/W
Preview selected	Enter or doubleclick

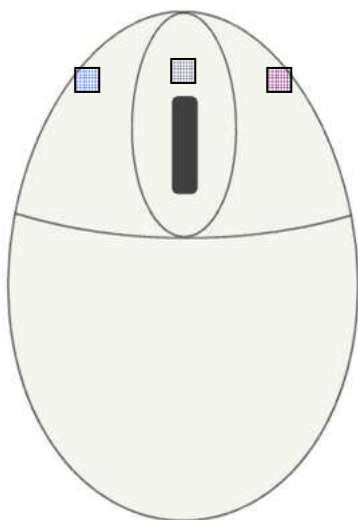
#### File Tools (when browsing the filesystem):




Compare selected object with...	=
Checksum and hash of selected objects	?

#### Modify:

Quick delete / Delete from archive	Del
Secure delete (files only)	Shift+Del
Refresh	F5 or icon in first column of titles' bar
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Cancel current selection and clear clipboard	Esc

### File manager's mouse controls



-  **Doubleclick:** preview selected object with associated application
-  **Rightclick:** activate file/archive browser's context menu;
-  **Middle button click:** extract selected object(s)

## Enter password (F9)

Password form allows to set password and, optionally, a keyfile for two factor authentication.

Once a password is entered, it is used by:

- file manager, since some encrypted archives needs password for being browsed
- archive extraction, to open encrypted archives
- archive creation, to create encrypted archive (if the selected format supports encryption); in archive creation interface it is specified if the password is set and if encryption is supported for selected archive format.

“Encrypt also filenames” option is used during creation of 7Z and ARC formats: if checked, the encrypted archive will need password for being browsed, else the content will be visible; in both cases extraction will require the password.



Image 9: enter password and keyfile

## Set advanced filters (F11)

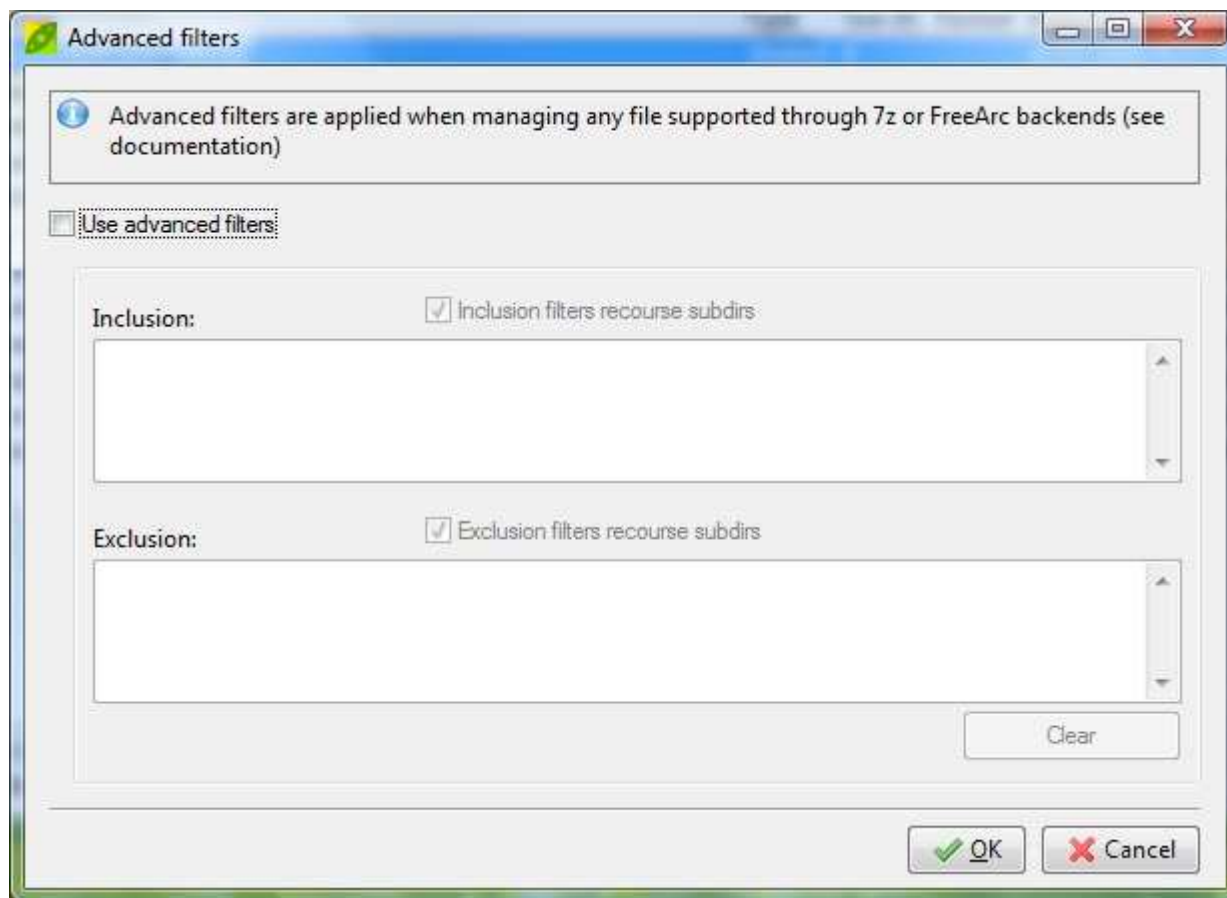


Image 10: inclusion and exclusion filters



“Advanced filters” form allows to use multiple inclusion and exclusion criteria, one per line, that can optionally recurse subdirs of the archive (“Inclusion/Exclusion filters recurse subdirs” options), and can be applied only to archive formats supported through 7z and FreeArc backends.

Advanced filters bypass the archive browser’s basic search filter (in mentioned archive types) so address bar is disabled if “Use advanced filters” option is checked.

Multiple filters, one per line, can be written in the inclusion and exclusion fields; string delimiters (“ on Windows and ‘ on Linux and other \*x systems) are not needed to be explicitly entered by the user.

In example, if the user needs to extract (or display) only “myfile.txt” plus all files named “your file” and all .mp3 files, but not .mp3 starting with a and m, could write in the inclusion field:

```
myfile.txt
your file.*
*.mp3
```

and in the exclusion field:

```
a*.mp3
m*.mp3
```

To exclude directories, use the syntax dirname\*\

Please refer to 7z documentation about inclusion and exclusion filters to understand how they works to get best result from this very flexible tool.

## Create keyfile (F12)

For higher security against dictionary and some social engineering attacks, a **keyfile** can be used along with the passphrase to key the encryption.

The keyfile need to be securely managed since its content need to remain secret as well as the passphrase; any file can be used as a key, but it’s strongly recommended to use a randomly generated file.

If a keyfile is used for a non-PEA archive, the SHA256 hash of the file (no size limit) encoded in Base64 (RFC 4648) will be prepended to the password, then it will be possible to work on archives encrypted with a keyfile using PeaZip or any application following the same convention, or simply entering the Base64-encoded hash as the first part of the password.

PeaZip can **create a random keyfile** sampling different entropy sources and submitting entropy collected to a robust random number generation routine, in the same interface it’s also possible to generate a 4-64 character **random password**; the password will contain mixed case base characters and digits only, in order to be typeable on any keyboard layout and to be accepted by almost all applications or online password forms.

The passwords and keyfiles generated in that way can be used not only in PeaZip but also in any other application requiring a strong password or a random keyfile.

This utility uses functions provided by pea’s libraries so refer to Pea documentation for any detail about random number generation and entropy collection in PeaZip project.

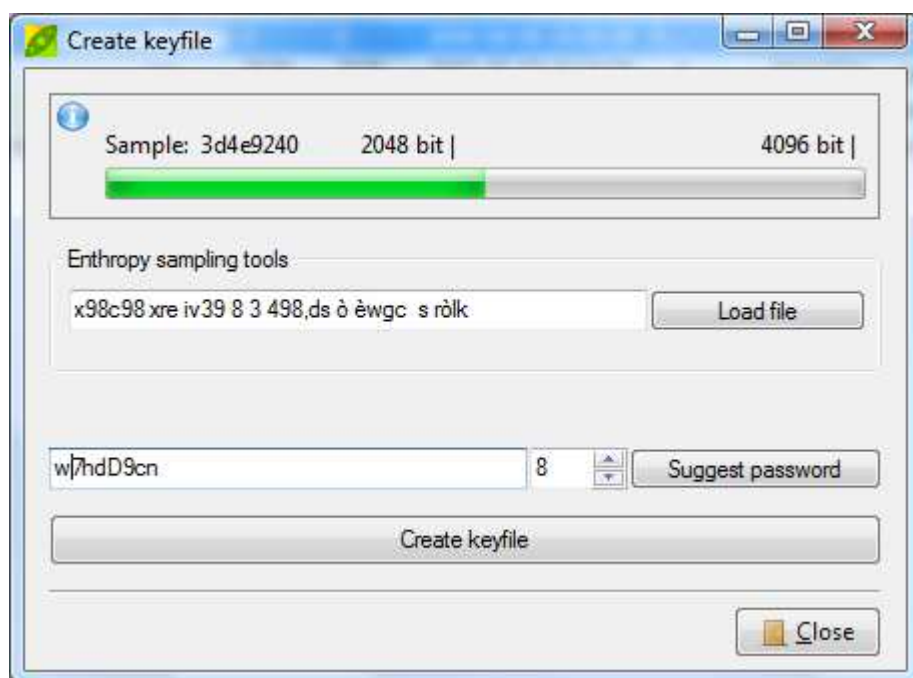


Image 11: keyfile generation utility

## File tools

PeaZip collect also handy file management utilities which are not strictly archiving-related.

Those utilities are accessible from file manager and (except for secure file deletion) from archive extraction and creation interfaces.

**Secure file deletion** is intended for securely remove files and folders from disk, avoiding possible data recovery.

In the application's context menus it is featured alongside quick deletion's entry rather than in other file tools' submenu.

Secure deletion accepts multiple files and directories as input, and provides multiple overwriting of file data with random data stream (AES256 CTR) forcing flush to disk each time, then replacement of content with randomly sized random data to fake file size, and multiple renaming of the file (or folder) with random string. Please use it carefully since wiped data will reasonably be not recoverable with known means.

Anyway please note that secure file deletion doesn't overcome any known risk of data leakage, since may exist copies of the data as temporary files saved by application that accessed the file, or as not securely deleted older version of the file, or cached by the system: wiping a file cannot affect that data, which can be recovered with software utilities or specific hardware probes.

Moreover, flash based storage usually re-allocates sectors for writing transparently for the software, in order to reduce unit's wear since flash units have a shorter lifespan in terms of writes; this doesn't allow to efficiently physically overwrite original content, reducing the efficiency of file wiping.

In those cases only wiping the whole disk would be effective, but this can be very time consuming and, for flash based disks, it will lead to fast wear and reduced lifespan of devices (complete disk wipe is currently not implemented PeaZip's file wipe procedure).

In "Settings > File Tools" it is possible to set number of passes to perform (2, 4, 8, 16) over the data.

**Compare files** performs byte to byte comparison between two files; unlike checksum or hash based comparison byte to byte comparison can spot exactly what are the different bytes and it is not susceptible of collisions under any circumstance, even if this condition is highly improbable and very difficult or not practically possible to trigger if a proper hash function is chosen.

**Check files** can perform multiple, user selected hash and checksum on multiple files at once; this is useful to find duplicate files and to check files for corruption when an original checksum or hash value is known.

In Tools > Settings > File Tools it is possible to select algorithms to be performed over the input files.

**File split/join:** plain split file is a format supported in "create archive" interface and split volumes can be merged back to original file from "browse archive" interface, but for ease of use those functions are available from "File tools" menu too.

**List/information** (from file browser): lists content of selected files/folders; in info mode it shows number of files and folders, older and newer object's date/time, total space occupation, and larger and smaller object's sizes.

**Hexadecimal preview (beta):** a very basic tool to view the content of a file represented as hexadecimal values.

Shows offset, hexadecimal representation of bytes, and possible utf8 translation of each string of 16 bytes per row.

At current level of implementation please note that:

- not all rows could be correctly represented in the GUI as utf8 strings, but they will be correctly written to file using save report feature;
- the implementation is slow and so it's limited to small files (up to 16MB).

# Extract archives

This interface is activated when one or more archives are selected for extraction from file manager, or when browsing an archive the whole content or part of it is selected for extraction.

The main application's menu is enabled (except for Browser submenu) and three tabs are featured.

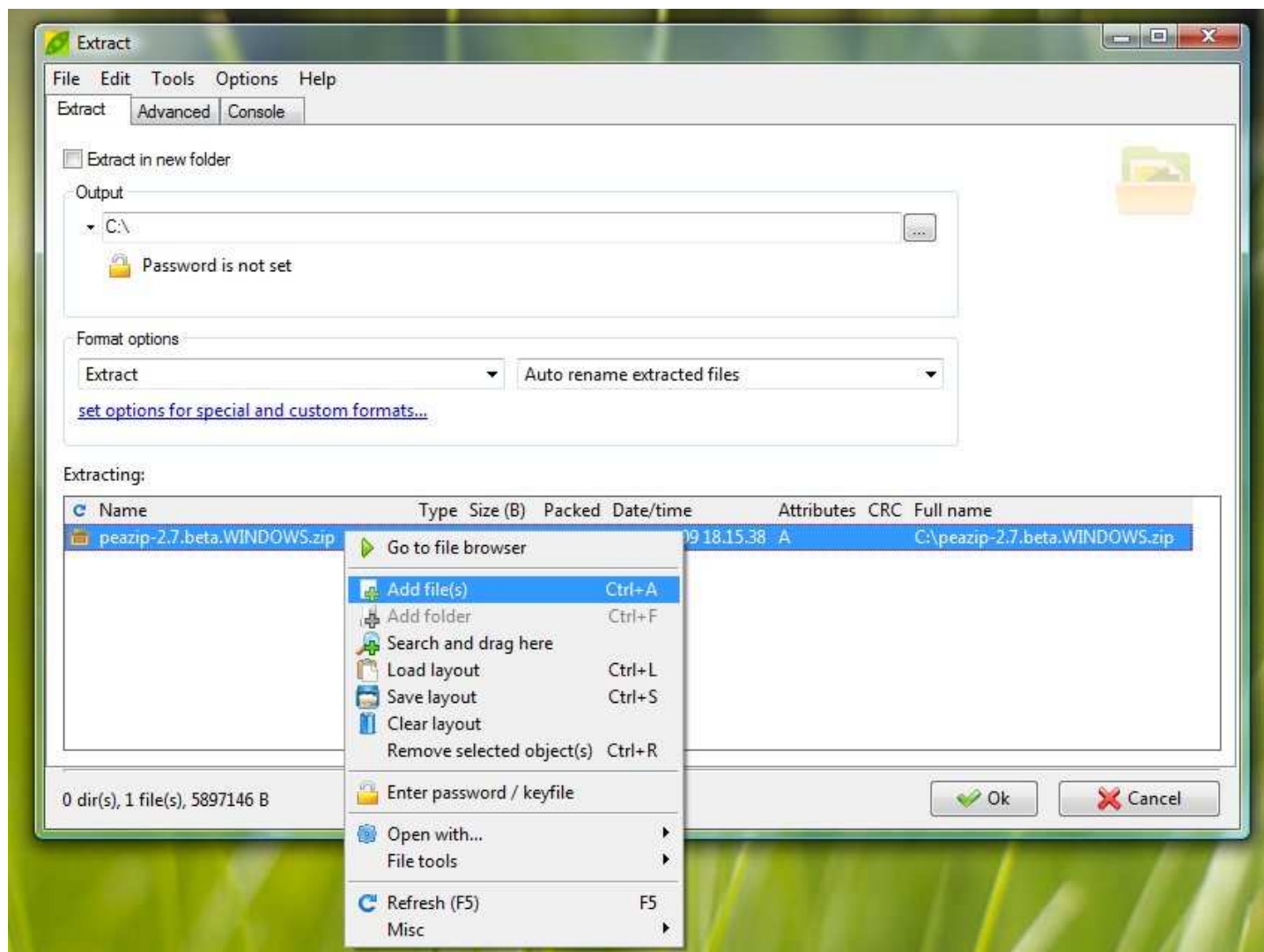


Image 12: Extraction interface showing context menu

## Extract

This is the main tab for extraction jobs, containing most commonly used parameters and **Ok** and **Cancel** button to confirm start or to discard the current operation.

**Extract in new folder** checkbox, when checked, triggers extraction to a fresh new folder, avoiding possible naming conflict and unwanted “tarbomb” effect (when an unexpectedly large number of files is extracted in current path getting mixed with existing files).

It is automatically checked when “Extract (in new folder)” is used instead of “Extract” menu entry.

**Output** group allows to select output directory; on the left of the address box, the arrow button popup a navigation menu similar to the one featured in file manager, to speed up the selection of most used paths.

The extraction's navigation menu is organized in **Filesystem**, **Bookmarks** and **Recent** submenus, containing only directory entries where the file manager's navigation menu can feature also file entries.

In Output group it is also possible to set the **password**, if needed for extraction, and to control if password is set or not.

## Format options

This group set options for 7z backend, that is used to support most common file formats.

- Possible actions performed by the extraction routine:
  - “Extract” extracts archived objects with paths, replicating the directory structure of the input data;

- “Extract (without path)” will extract all archived files to the same path;
- “List” will show archive’s content;
- “List (with details)” will give a more detailed report on archive’s content, the same given by “Info” entries in context menu. List functions will always be performed in pipe mode (even if 7z option is set to “console mode”), using graphical wrapper in order to make easier reading and saving the report.
- “Test” will perform type specific tests to prove or disprove archive’s integrity.
- What the extraction routine will do in case of naming conflict while extracting data (when changed, this parameter is saved to PeaZip’s configuration file):
  - “Auto rename extracted files” assign a new unique name to objects being extracted from the archive each time a naming conflict is encountered; that policy assures that pre-existing objects will keep their names and new ones will get new unique names (default).
  - “Auto rename existing files” assures that extracted objects get the desired name while pre-existing objects are renamed with a new unique name.
  - “Overwrite existing files” make all pre-existing objects overwritten by extracted objects.
  - “Skip existing files” assure that pre-existing objects are not touched by the extraction operation, being the conflicting objects not extracted from the archive.

This group also links to “Advanced” tab which contains job parameters for other backend executables.

On the bottom of the tab are listed the archives that are going to be extracted (**extraction layout**), with the total number of archives and total size; additional archives can be dragged here to be added to the list.

Rightclicking on the input list shows the **contextual menu**, which features functions to add archives to extraction layout (add files, search and drag here, load layout), and other related functions.

The layout can be saved to a UTF-8 text file (for maximum flexibility of use); when a layout is loaded each object is checked (must exist, duplicates are skipped).

From the context menu it is also possible to remove objects from the archive’s layout (“Remove selected objects” and “Clear layout”) and to explore selected object’s path.

“Go to file browser” can be used to return to file browsing without discarding the current list of archives, in example to navigate and search for other archives to be added to the list from file browser interface.

“Open with...” submenu of context menu allows opening the selected object with PeaZip, associated application, or a custom applications.

“File Tools” submenu allows quick access to some PeaZip functions to be applied on selected objects (see “File tools” chapter).

Objects in the archive layout can be sorted by name, full name, size, extension, type, attributes etc, clicking on titles in archive layout’s title bar.

## Advanced

This tab features “**Special formats**” group which is similar to “Format options” group of previous tab, and allows to set parameters for non-7z backend executables. Notably, in this group it is possible to change action for FreeArc backend to “Repair”, which (for ARC archives only) verifies integrity and tries to repair the archive(s) using the recovery records that may have been included at the archive creation.

**Extract supported non-archive types** checkbox, unchecked by default, allows some non-archive types like executables, MS Office and Open Office documents etc to be treated as archives, in order to be disassembled.

In file manager those file types are displayed with special icons to show they can be accessed as archives by PeaZip, but will not be treated as archives by default.

**Extract unsupported file types** checkbox, unchecked by default, allows any arbitrary type of file to be treated as an archive by PeaZip, providing the user can specify a custom executable backend to handle the file type in “**Custom parameters**” group, which is enabled if this option is checked.

## Console

From this tab, only when extraction is launched while browsing inside an archive, it is possible to transform the job defined in the GUI interface into a command line that can be edited (independently from the job definition in the GUI frontend).

Rightclicking on “Click to import...” label or on “Save job” button it is possible to compose the command line operating on all, displayed or selected objects, if partial extraction of content is supported for the current archive type (7z, arc, rar, tar, zip...).

This command line can be launched or saved as a text file for future use, as study, scripting, analysis etc.

## Create archives

This interface is activated when files and folders are selected for being added to an archive from file manager, or when, browsing an archive, it is requested to update it adding other files and folders.

The main application's menu is enabled (except for Browser submenu) and three tabs are featured.

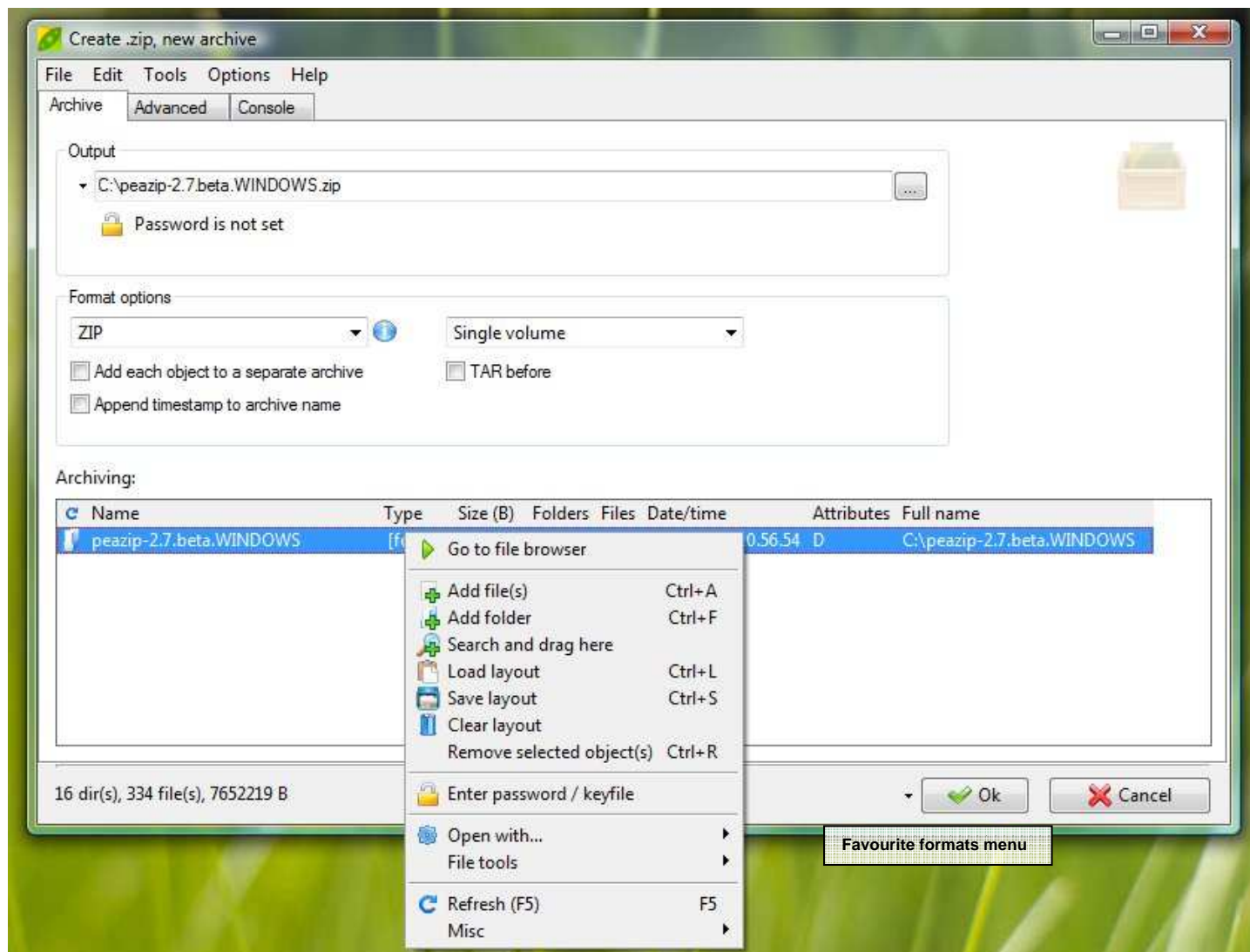


Image 13: "Create archive" interface (on Windows Vista)

## Archive

This is the main tab for archive creation (or update) jobs, containing most commonly used parameters and **Ok** and **Cancel** button to confirm start or to discard the current operation.

**Output** group allows to select output directory and file name; on the left of the address box, the arrow button popup a navigation menu similar to the one featured in file manager, to speed up the selection of most used paths.

The archiving navigation menu is organized in **Filesystem**, **Bookmarks** and **Recent** submenus, containing only directory entries where the file manager's navigation menu can feature also file entries.

In Output group it is also possible to set the **password** to create an encrypted archive; if a password is set the field will inform if the selected archive format supports encryption.

From password popup it is possible to check/uncheck "Encrypt filenames" option, which, if supported by the selected archive format (7Z and ARC formats) does not allow to browse the content of the archive if the right password is not provided.

**Format options** group allows to set the archive format, more options for each format can be fine-tuned in "Advanced" tab (see next chapter for more details); last used compression options will be remembered to be used as default, avoiding the need to set it in advanced tab following times the format is used; resetting applications default resets also compression options.

The info icon in the right of the archive type selection combobox will display a brief explanation of the characteristics and features offered by the selected format.



The arrow on the left of OK button shows **favourite formats** menu, which for brevity and simplicity displays only some of the supported archive format, the most widespread. Favourite formats can be customized in Options > Settings > Create archive.

In the same group it is possible to set **Volume size**, allow (optionally) splitting the resulting archive in volumes of given size, choosing between presets (from 1.44 MB FD to 8.5 GB DVD DL size) or freely composing a custom size. This option is not supported by some formats (i.e. self extracting archives) and will be automatically overridden if not supported (the combo box will be greyed to be noticed by the user).

*Hint: to store large files/archives on **small supports**, or to respect **mail attachments size limitations**, it is possible to split the output in volumes of desired size ("Options" tab), instead of spending more time for a deeper compression which is not always capable to reduce the output under the desired size.*

**Add each object to a separate archive** send each item in the list to a separate archive.

In example, adding x folders to the archive list with this option checked will create x separate archives; if it is desired to add each file contained in a folder to a separate archive, instead, select the files in file manager and click "Add" to add them individually to the list.

**TAR before** option allows to consolidate all input objects in a single TAR archive, temporarily saved in the output path, which will then (in a second pass) be compress/encrypt/split using the format specified in archive type combo box (after that second pass the temporary TAR archive will be deleted).

That option allows easily merging the advantages of TAR format (of mainstream and standard usage on most Unix systems) with features of other archive formats. It is especially useful as it allows to select compression-only formats (like gz, bz, quad) to compress archives of multiple objects (resulting i.e. in tar.gz or tar.bz or tar.quad) quite transparently for the user, but however it can be used in conjunction with any format (forming i.e tar.7z, tar.paq, tar.pea and so on).

The application will try to check if "TAR before" option is needed; however the user can check/uncheck this option anytime before launching the archiving process.

**Append timestamp to archive name** will append current date and time to the name of the archive, it is useful for archiving and backup purpose.

On the bottom of the tab are listed the archives that are going to be archived (**archiving layout**), with the total number of files and folders and total size; additional files and folders can be dragged here to be added to the list.

Rightclicking on the input list shows the **contextual menu**, which features functions to add files and folders to archiving layout (add files, add folder, search ad drag here, load layout), and other related functions.

The layout can be saved to a UTF-8 text file (for maximum flexibility of use); when a layout is loaded each object is checked (must exist, duplicates are skipped).

From the context menu it is also possible to remove objects from the archive's layout ("Remove selected objects" and "Clear layout") and to explore selected object's path.

"Go to file browser" can be used to return to file browsing without discarding the current list of archives, in example to navigate and search for other files and folders to be added to the list from file browser interface.

"Open with..." submenu of context menu allows opening the selected object with PeaZip, associated application, or a custom applications.

"File Tools" submenu allows quick access to some PeaZip functions to be applied on selected objects (see "File tools" chapter).

Objects in the archive layout can be sorted by name, full name, size, extension, type, attributes etc, clicking on titles in archive layout's title bar.

## Advanced

This tab contains fine-grained options for each supported format, including compression level.

Once advanced options for a given format are set, they are remembered, so it is usually not needed to use Advanced tab and all of archive creation can be done form Archive tab.

*Hint: most of **multimedia file formats** (like jpg, png, mpg, avi, mp3...) are already compressed with algorithms that are either lossy and/or strongly optimized for the specific data structure, so compressing them with general purpose lossless compression algorithms, even most powerful ones, are likely to provide only marginal benefits in terms of size, compared with huge benefits which can be obtained compressing other types of files (bmp, tiff, doc, xls, txt, html...).*

*Consequently, it is recommended to use fastest compression settings, or fast algorithms (i.e. gz and zip's deflate), or even "Store" option to don't compress, to archive those types of files in a computationally efficient way; in this way it is possible to consolidate and optionally encrypt files that are sent to the archive, without spending much time for compression.*

Backend executables can be used by PeaZip in the native console mode or, by default, through a graphic wrapper (see "PeaLauncher" chapter), displaying additional information usually not given by the console application (exit code

explanation, input size, elapsed time in ms, speed in B/ms) and allowing to save a job report (console output plus additional information); this behaviour can be customized in Options > Settings.

**7z** - *Performances scales very well on multicore machines; following formats are recommended:*

*7Z when high compression is desired; offers powerful encryption*

*ZIP to provide archives which all Windows user can read with integrated "compressed folders" utility, or with most of mainstream file/archive managers*

*TAR (optionally compressed with GZ or BZip2) to provide archives which most Unix (Linux, BSD ...) users can read with applications usually bundled by default*

Selecting a format supported through 7z executable (7z, Bzip2, GZip, Self-extracting, Tar, Zip) it will be displayed the 7z option subpanel, featuring Options, Compression and Encryption group box.

"Compression" group box allows choosing compression level and algorithm and to fine tune compression options, which are format specific.

*Last used compression level and method is remembered by PeaZip for 7z backend, but other custom options (dictionary, word, passes, solid block size) will be remembered only for the current session (until the archive type is changed or edited) and next times the last used compression level will be proposed with its default settings.*

Option "Compress files open for writing" allows to add files to the archive even if open for writing by other applications, very useful when running backup jobs (otherwise, if the option is not checked, open for writing files will be skipped); the last used setting is remembered.

"Create self-extracting archive" options create a Win32 executable (.exe) that will self extract archive content (archived and compressed in 7z format); the receiver will then not need any application to extract the archive since all what is needed for extraction is embedded in the archive itself. As drawback the resulting file will be about 80KB bigger than the raw archive and the executable must be in a single volume (Volume size option will be greyed). Checking this option the user can choose between a console and a graphical interface (default) for the self-extracting application.

"Encryption" group box contains encryption related options:

- "Algorithm" allows to chose encryption algorithm; 7Z format supports AES, while ZIP supports AES and ZipCrypto algorithm. AES is always used with 256 bit keys. ZipCrypto is a weak algorithm but may be useful if the user need to generate encrypted .zip archives compatible with some outdated applications not supporting new WinZip standard AES-based AE encryption.

"Options" group box allows choosing other format specific options:

- "Function" combo box allow to choose the policy to follow when a pre-existing archive with the given name is found on the system:
  - new archive will force the creation of a new archive, with unique name;
  - add will append the input objects to the archive content, if an archive with the given name exists;
  - update will substitute matching objects into the archive, if existing, with input objects.
- "Threads" combo box allow specifying the number of threads to try to generate for parallelising and speeding up the execution of the application (possible only for LZMA, Deflate, Deflate64 and BZip2 algorithms) on Hyper threading-enabled / multi-core / multi-CPU environments; on Windows systems single processor systems will be recognized and will use "no multithreading" option by default while multi processor system will use "generic multithreading" option by default; on non Windows systems "no multithreading" is the default option.
- "Other" edit box allows to freely enter additional parameters for the archiving job. This string is inserted by default after all the parameters set by the GUI and its syntax is not checked, so use with caution.
- Send by mail open a new mail in the default mail client, attaching the resulting archive; it requires a compatible mail client as default system client, i.e. Outlook or Outlook Express in Windows, and doesn't work with multi volume archives (the options is hidden if Volume size is not "Single volume").

**ARC** - *Recommended when high compression is desired; offers powerful encryption and, optionally, recovery records. Performances scales very well on multicore machines.*

In FreeARC's options subpanel, it is possible to adjust compression level, specify file grouping strategy for solid archives, create recovery records to attempt archive's repair in case of corruption, encrypt the archive with various encryption algorithms (AES, Serpent, Twofish, and Blowfish), create self extracting archives selecting from many sfx modules. The format is currently being actively developed.

**Custom** - *Allows to select an external compressor/decompressor to support file types which are unknown to PeaZip*

Selecting "Custom" format, it will be displayed a subpanel allowing to specify archiver/compression utility to use to perform the job, alongside parameters (free editing) and syntax (the way parameters, input list and output name should be organized on the resulting command line).

Last 8 used custom executables are remembered and can be chosen from a popup menu rightclicking on executable's selection control.

Please note that exact syntax of the command for a custom executable may need to be utterly adjusted, this can be done in "Console" tab, which allows importing and free editing of the job definition.

**Pea** - *Recommended when powerful encryption and integrity check are desired; provides fast compression*

Selecting Pea as archive format it will be displayed the Pea option panel; this is the archive format developed from the author of PeaZip and it is supported through Pea executable; for more information on it you may look the documentation about Pea on <http://peazip.sourceforge.net/peazip-help.html>.

**QUAD/BALZ** - *Recommended when it is desired to provide average to good compression, with fast uncompressing*

Selecting QUAD/BALZ, the option subpanel will allow to choose compressor engine between QUAD and BALZ (newer ROLZ-based compressor), and if using max compression mode (for both QUAD and BALZ, compression will be slower, but decompression will remain fast). QUAD and BALZ are compression only algorithms and likewise other ones (GZ, BZ2, LPAQ) they can benefit of “TAR before” switch for handling multiple input files.

**Split** - *Recommended to split a single large file to the desired size, without attempting compression*

Options for file split are limited to optional integrity check algorithm to be performed on the file; integrity check information will be saved on a separate file, allowing files splitted by PeaZip to be joined by other similar applications (like Unix split, Hjsplit, FileTools etc).

**UPX** - *Recommended to developers to reduce the size of executables*

UPX compression is intended mainly for developers needing to reduce the size of executables before distribution.

It can accept only a single executable file at time and is not intended to be used as a general purpose compression utility; in fact misapplying Strip and/or UPX on non suitable executables (i.e. jet stripped executables) may easily lead to unusable output executables.

It is possible to omit either UPX compression (selecting “do not compress” in “Compression” combo box) or Strip pre-processing (unchecking “Strip before UPX”).

By default it is created a backup copy of the executable before Strip/UPX (option “Keep executable’s backup”), named as the executable with .backup extension appended.

**\*PAQ** - *Recommended when highest possible compression is desired (experimental; speed and memory usage makes it not recommendable for general purpose use on current generation machines)*

PAQ is a very powerful compression scheme and it’s presently in research state; different versions and branches exist and you should use the very same implementation to compress and uncompress data, PeaZip uses PAQ8O for compression.

It brings unmatched compression ratio, better than any other compressor; the downside is that the algorithm has very high memory and computing power requirements for today’s machines, resulting to be very slow if compared to mainstream compression algorithms, so the user should carefully consider if the speed/compression trade-off would be advantageous case by case.

To obtain best results while compressing many small files you should also consider consolidation them before (i.e. using tar) since PAQ would store filenames (and sizes) in uncompressed form.

ZPAQ is faster and lighter than PAQ, at the cost of a slightly inferior compression ratio.

Please note that for ZPAQ currently PeaZip supports only storing/extracting to full pathnames, so output options will be ignored for that format.

## Console

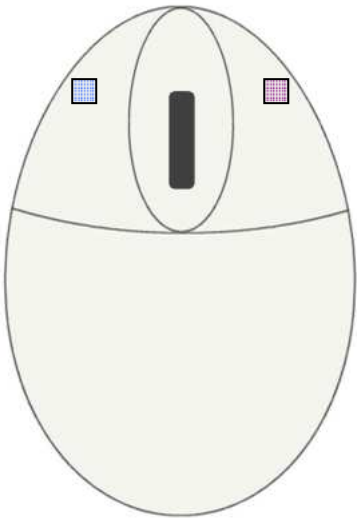
From this tab it is possible to transform the job defined in the GUI interface into a command line that can be edited (independently from the job definition in the GUI frontend); this command line can be launched or saved to a text file for future use (study, scripting, analysis etc).

Keyboard shortcuts for archive extraction and creation

~ `	1	@ 2	# 3	\$ 4	% 5	^ 6	& 7	* 8	( 9	) 0	- _	+ =	← Backspace
Tab ↔	Q	W	E	R	T	Y	U	I	O	P	{ [	} ]	 _
Caps Lock ⬆	A	S	D	F	G	H	J	K	L	: ;	" '	↵ Enter	
Shift ⬆	Z	X	C	V	B	N	M	< ,	> .	? /	Shift ⬆		
Ctrl	Win Key	Alt								Alt	Win Key	Menu	Ctrl

<b>File tools:</b>		
Checksum/hash selected files	?	
Compared selected file with...	=	
<b>Archive layout-related:</b>		
Add file(s)	Ctrl+A	
Add folder	Ctrl+F	
Add from search dialog (drag to archive)	(context menu only)	
Load archive's layout	Ctrl+L	
Save archive's layout	Ctrl+S	
Open object with default application	Ctrl+O or Enter or doubleclick	
Open object with ...	Ctrl+W	
Explore object's path	Ctrl+E	
Remove selected object from archive's layout	Cancel or Ctrl+R or Ctrl+Backspace	
Refresh	F5 or refresh icon on the left of layout's titles row	

Mouse controls for archive extraction and creation



- Doubleclick:** open selected object with associated application or browse folder
- Rightclick:** activate “create layout” context menu;

# PeaLauncher

PeaLauncher is the job's graphic wrapper, a component that graphically displays an underlying job performed by a console-based backend used by PeaZip to create, extract, test or list an archive

By default, the window closes if no errors are detected, but remains always open for test, info and list jobs which requires the user to read the job's report.

*It is possible to always keep the job's window open after job termination, to inspect job's log and command line, selecting "Always keep open" in PeaLauncher combo box in Options > Settings*

*It is possible to automatically open the output path (where the archive was created, or extracted) checking "Open output path when job completes" in Options > Settings*

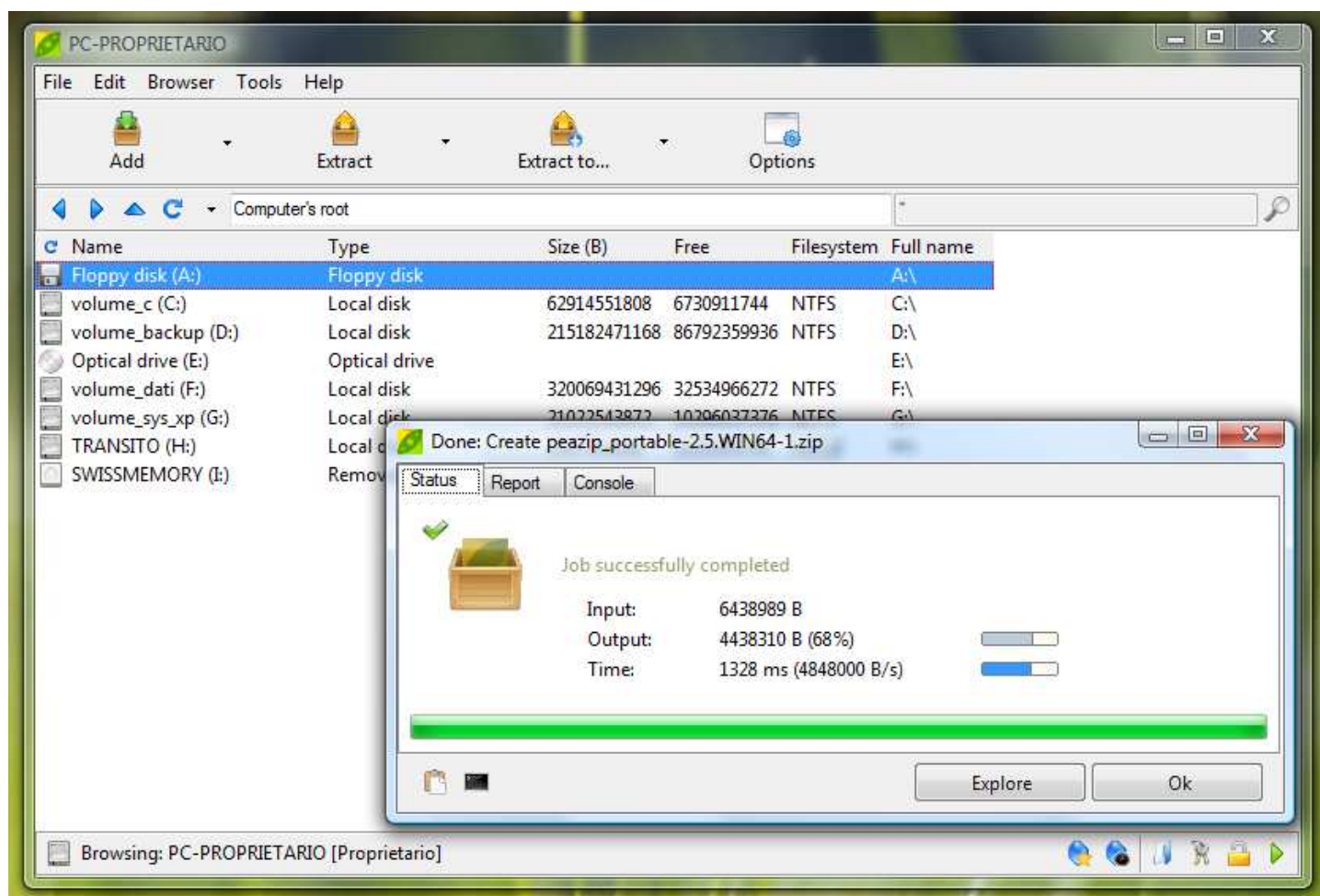


Image 14: PeaLauncher

The graphical wrapper is not invoked if jobs are running in console interface, either because they need to run in console (UNACE, UPX) or because the user selected to use "Console interface" in program's options; however list and test always run in graphic wrapper mode using pipes to give readable job log.

In console mode, the console will automatically close when job is completed without prompting any message if the job completes without errors, otherwise an error report will popup.

PeaLauncher can run also in "GUI + console" mode (see Settings chapter) displaying both the graphical interface and the underlying job running in its native console interface, showing its native progress indicator, usually more detailed and reliable than GUI's one.

PeaLauncher can be invoked with no parameters, or with a single archive file as parameter (i.e. dragging an archive over the executable, or a link to PeaLauncher), to be used as a minimalist standalone extraction application.



# Settings

Options > Settings allows to configure behaviour and aspect of PeaZip application.

## General (1)

- “Localization” set application’s language; clicking on the “...” button it can be chosen a language file; language files are stored in PeaZip’s /res/lang subpath (which can be explored clicking on the language file string). If the language file is correctly loaded, translator’s name and translation’s revision date will be displayed and the application will be restarted with chosen localization.
- “Desktop” contains the path of user’s desktop, if correctly got from PeaZip; it also allow to set an arbitrary path to be used as desktop for the application, if preferred.
- “Backend binaries user interface” group allow to chose the way the backend command-line applications will run:
  - in the native console interface (console mode), giving detailed and real time progress indication;
  - through a graphical wrapper using pipes (graphic, default), giving a very detailed job log and allowing to pause, resume and change priority of the job;
  - graphic + console: like the previous, but also showing native console interface: gives a responsive UI and at the same time plenty details and real time progress indication.
- “PeaLauncher” sets the policy about PeaLauncher window after job termination:
  - always keep the window open (as in previous versions), useful to inspect job’s report and command line
  - keep open only if needed (default from version 2.6), keeps the window open only in case of errors, or for test, info and list jobs requiring the user to read the job’s report
  - always close the window at job termination, regardless the kind and the result of the job
- “Open output path when job completes” if checked opens the output path (where the archive was created, or extracted) after the job is completed, like some archivers does.

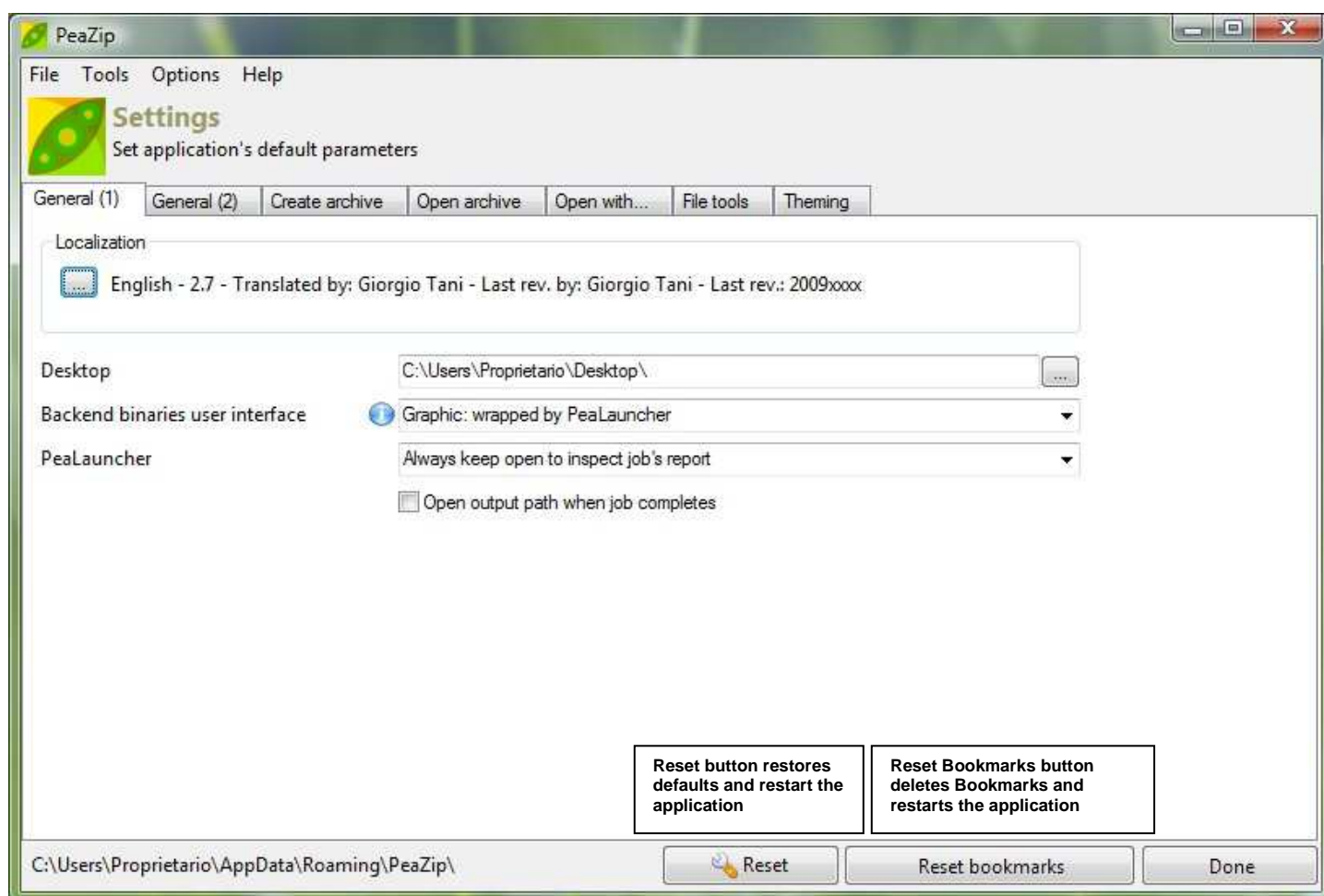


Image 15: default options panels

## General (2)

- “Encoding” group contains character encoding related options
  - Encode job definition as UTF-8 text: if checked job definition files are saved with UTF-8 encoding header, otherwise no header will be prepended to the file.
  - Archive browser interface’s character encoding option: if flagged, displays extended characters in archived object’s names as UTF-8 text, otherwise replaces extended characters with “?” jolly character (as in pre-2.5 releases).

Please note that conversion to UTF-8 text works fine on Linux (if UTF-8 is the system's text encoding, as usually is) but on Windows, at current state of development, the conversion is successful if system's console environment uses single-character encoding or UTF-8 (CP65001).

*Note: replacing extended characters with "?" jolly character may improve commands syntax if, for any reason (i.e. limitations due to guest OS, archiving software or archive format), the character set used while creating the archive cannot be successfully translated on the current machine opening the archive.*

- "Other" group:
  - "Save history of last used archives": on by default, allows to keep track of recently created and opened archives (File > Recent submenu); disabling the option the history will not be tracked, the submenu will be disabled and existing items will be cleaned closing the application.
  - "Save main window state", on by default, keeps the windows size and position on exit, otherwise last used size and position is kept, ignoring utter modifications.
  - "Show hints", on by default, enable or disable application's hint popup.

### Create archive

- "Favourite formats" allows choosing a selection of user's favourite formats to be displayed in the popup menu on the left of Ok button in archive creation interface, for a quicker selection.
- "Default format" sets the default archive/compression type, by default last used format is kept

### Open archive

- "Archive browser interface" sets the way the content of archives is displayed: a classic navigable browser (default), or flat view to show all archive's content, or last used mode.
- "Fast open routine, stop browsing if list exceeds:": default on. PeaZip will stop representing archive's content in the GUI if the resulting list's memory stream is bigger than the given value. It allows to quickly opening very big and populated archives stopping preliminary operations if they are taking too much time/memory, allowing the user to narrow the selection with full functional search or filter functions. Extract/list/test operations are not affected by this setting, it is applied to archive browsing only.
- "Advanced" group contains less often used settings
  - "On extract/list/test operations from system's menus" set PeaZip behaviour when an extract, list or test operation is launched from a system's menu entry:
    - Don't ask for password: assume the archive is not encrypted and immediately start the operation without spending time in searching if a password is needed;
    - Test if password is needed (slower): the default behaviour with PeaZip evaluating if a password is needed, time needed depends from type and size of the archive;
    - Always ask for password: always popup password request, note that it is possible to ignore it (press enter or Ok button) to immediately start without providing a password.
  - "Always ignore paths for Extract displayed" and "Always ignore paths for Extract selected": off by default; ignore paths extracts to output path not re-creating the directory structure as is in the archive; applies to extract and extract to functions.
  - "Always ignore paths for Extract and...": on by default, works as the previous items on object extracted through extract and... functions.

*Hint: when extracting a folder from the archive the paths will be always preserved, overriding this option for all ignore paths switches.*

### Open with...

- "Custom editors, players, antivirus scanners" provide two sets of up to 8 applications (or scripts, or commands) to be used to open files overriding system's file associations. Applications can be sorted dragging up or down the application's number in the list; rightclick to select, edit or remove applications, and enter descriptions. Some commonly featured applications are preset and can be recalled with "Reset" link; the scripts are saved to a separate configuration file "custedit.txt" and are not cancelled with application's reset (as well as "Bookmarks" list).
  - "Basic edit" set allows to edit custom application with ease, selecting or typing and application or command to be used to open the file. It is also possible to enter parameters after command or application name; after the "Executable or command" string it will be passed a space and the input file name.
  - "Advanced edit" set allows instead a bit more complex syntax, providing a string to be entered before and after the input file name (it's up to the user decide if spaces between strings and file name are needed).

*Note: by default antivirus / antimalware scanners are defined in "Advanced edit" set, because some of them requires a bit complex syntax, but it is only a convention.*

*Note: syntax's examples of preset applications can be used as model to start customizing entries in both sets; clicking an entry in "Advanced edit" set will show a line displaying complete command entered, with the pseudocode "%f" representing the input file name position in the command string.*

## File Tools

- “Checksum/hash files” group allows to select algorithms to be used for file checksum/has and how to display the result (hexadecimal, hex LSB, Base64).
- “Secure deletion” group allows choosing number of passes to perform to securely delete files (2-16); each pass perform: overwrite file with random data, mask original file size, rename file.

**Theming** tab allows to change the application colours and icons and, under Win32 (from Windows 2000), opacity.

## Customize theming

At start-up the application loads a configuration file in “res” directory in the application’s path, loading theming parameters along with other configuration variables; theming parameters are applied to PeaZip, Pea and PeaLauncher graphical applications.

From the “Theme” drop down menu the user can chose to use preset themes which comes packed with PeaZip, or a custom theme (choosing the “theme.txt” file from /res/themes/themename path)

From version 2.5.1 PeaZip supports compressed themes: a theme can be stored as zip or 7z file, to be extracted only if selected by the user.

Theme’s settings set the default values for: **icons folder**, **application’s colours**, **browser’s row height**, **GUI item’s height** (auto sized by default) to i.e. adapt the application to larger fonts, **adapt toolbar buttons** (i.e. to fit longer text in some languages), and on Windows **opacity** value.

Theming values can be saved in the current configuration, and will override correspondent theme settings, which can be restored clicking on “Reset” link for each variable.

To prevent using of too high levels of transparency that will make the application unusable (up to totally invisible) there is a maximum level of transparency hard coded for the application.

The user can also create new themes from current settings, choosing a theme name and path: theming variables will be saved as a new theme file; some themes come with the application, in “themes” dir under “res” path. Theme files (and configuration file) can also be manually edited with a text editor.

*Hint: it is also possible to edit theme graphic in theme’s path, the new bitmap will be loaded at program startup*

Theme folders should be saved in /res/themes path; default theme, FireCrystal, should not be removed, since in case of theming failure it allows to roll back automatically to basic applications colours and icons (Pea executable, meant to be used not only through PeaZip, has hard coded theming values to not need relying on PeaZip’s conf and theme files).

From version 1.8 transparency is not activated by default, being the default opacity set to 100%, which make the transparency-related code not used and allowing better UI performances; to use transparency simply go to theming panel and drag the transparency bar to the desired level.

**Program’s icons**, on Windows, are stored in \PeaZip\res\icons\ path and can be edited with a suitable editor or replaced with custom icon files; on Linux systems icons (in PNG format) are stored accordingly the distribution and the desktop environment policies, i.e. /opt/kde3/share/icons\ or /usr/share/icons\

*Hint: custom, user-provided icons and other resources can be found on “PeaZip resources (misc)” download group on PeaZip project on SourceForge, and on PeaZip’s website.*

## Advanced settings editing

Program’s configuration is stored in conf.txt file, which by default is saved in /res folder in application’s path, but an alternative location, either as absolute or relative path, can be set in second line of altconf.txt file in the same folder: “**same**” string in altconf.txt specify the data has to be saved in program’s res path (best for portable packages) and “**appdata**” specify the data has to be saved to %appdata%\PeaZip\ path in user profile (Windows) or /.PeaZip/ path in user’s home on Unix systems, to guarantee write access to data from current user’s profile and allow different users to store different and private profiles for PeaZip.

In this way only the invariant data (including altconf.txt, binaries, graphic and themes, language files etc), which needs to be accessed only for reading during normal program’s usage, is stored in program’s path, that could be even set as read-only.

All variable data files (conf.txt, rnd, custedit.txt and bookmarks.txt), which can need to be accessed for writing during normal program’s usage, will be stored in res folder (by default) or any other path specified in altconf.txt.

Variable data path in use is shown on the bottom of the Settings panel; if the altconf.txt file is deleted PeaZip will rebuild a default one which sets variable data path in /res folder.

The user will then be able to keep separate pre-set configuration files which can be used alternatively editing the altconf.txt file, or to package the application to fulfill custom needs as to keep the variable and invariant data stored in separated paths.

# Supported file types

See also <http://peazip.sourceforge.net/peazip-free-archiver.html>

Main families are:

Through **Pea** executable (LGPL, Windows and Linux)

- Full support
  - PEA: security focused, flexible integrity check and optional two factor authentication with passphrase and keyfile (AES256 EAX mode authenticated encryption), fast compression comparable with Zip/Gzip, native multi volume spanning.
  - Split: compatible volume spanning (file split/join) function with optional integrity check

Through Igor Pavlov's **7z** (LGPL, Windows) and Myspace's POSIX 7z (LGPL, Linux)

- Full support
  - 7z, 7z sfx: feature-rich archiving format, strong AES encryption, awesome compression ratio and optionally auto extracting archives (sfx, Win32 executables)
  - Bzip2 (BZ, BZ2, TBZ, TBZ2): single file compressor, adequate speed and good compression ratio
  - Gzip (GZ, TGZ): fast single file compressor, adequate compression ratio
  - TAR: mainstream archiving and backup format for Unix platforms, often used to archive data to be compressed with GZip, BZip2 or other algorithms
  - ZIP: mainstream archiving and compression format for Windows platform; support covers also Deflate64-compressed archives and AES-encrypted archives
- Browse/extract support
  - ARJ, LHA, LZH: popular archiving format on DOS and early Windows platform
  - CPIO, Z, TAZ, TZ: archive/compression formats for Unix platforms
  - LZMA: single file compressed with LZMA algorithm, introduced with 7z format; and XZ: single file compressor, based on LZMA-family algorithm, providing excellent compression ratio
  - RAR: popular archiving and compression format, with advanced encryption and error recovery features
  - Various archive types based on ZIP or its modifications
    - Java archives: JAR, EAR, WAR
    - PAK, PK3, PK4 : modified zip archives, used to store data by some popular games (like Quake3, Quake4, Doom3)
    - SMZIP
    - UP3: U3 portable application's package format
    - XPI: Mozilla package format (addons for Firefox, Thunderbird etc)
  - Various disk images
    - ISO standard disk image format
    - UDF widespread disk image format
  - Various executable file types
    - NSIS: Open Source Windows installer format
    - Some Windows executables
    - MSI Microsoft's proprietary installer format for Windows
  - Various (non-archive) compressed file formats, like
    - CAB compressed archive format
    - Compound
      - Some of MS Office formats: DOC, XLS, PPT
    - CHM, CHW, HXS compressed help files
    - WIM, SWM Windows image format
    - OpenOffice file types: container files for text, database, image and multimedia data: ods, ots, odb, odf, odg, otg, odp, otp, odt, ott, oth, odm, oxt.
    - Gnumeric .gnm files
    - SWF, FLV
  - Various Linux installer formats:
    - DEB (Debian based)
    - PET/PUP (Puppy Linux)
    - RPM (Redhat based)
    - SLP (Stampede Linux)
  - Various Macintosh formats: DMG/HFS package/disk image format

Through **PAQ/LPAQ**, Matt Mahoney et al. (GPL, Windows and Linux)

- Full support
  - PAQ80, ZPAQ
- Read only

- Older PAQ8 formats F/JD/L: experimental compressor; at cost of high computing time and memory usage provides best known compression ratio for most data structures
- LPAQ1/5/8: lighter and faster version of PAQ, at the cost of some compression (single file compressor)

Through **Strip** (GNU binutils) and **UPX** (GPL Markus F.X.J. Oberhumer, László Molnár and John F. Reiser)

- Compression only
  - Strip strips symbols from executables and UPX apply compression: this allows containing the size of binaries of different types (exe, elf, etc...), which is important i.e. for distribute smaller packages; this feature is mainly oriented to developers.

Through Ilia Muraviev's **QUAD** (GPL) and **BALZ** (Public Domain)

- Full support
  - QUAD: high performance ROLZ-based compressor which features high compression ratio and fast decompression
  - BALZ: enhances overall performances compared to QUAD

Through Bulat Ziganshin's **FreeARC** (GPL)

- Full support
  - FreeArc's ARC/WRC, FreeArc's sfx: experimental archive format, featuring efficient compression (high ratio and good speed), and advanced features like strong encryption and recovery records

### (SEPARATE PLUGIN)

Through UNACEV2.DLL 2.6.0.0 (Windows) and UNACE (Linux): royalty-free, Marcel Lemke, ACE Compression Software

- Browse/extract support
  - ACE: popular compression format, used mainly on Windows systems



## Customisation and scripting

PeaZip portable doesn't need installation and doesn't modify the host system, however program's most used functions can be integrated, under Windows, in SendTo and context menu and, under Linux, in FreeDesktop-compliant desktop environments through .desktop files (Gnome, KDE...) and Nautilus scripts (Gnome); examples are in FreeDesktop\_integration folder included in Linux packages.

In the same way it is possible to extend the integration automatically provided by installable packages, creating quick links in system's menus or in scripts for most of the program's internal functions.

Quick access to most used PeaZip's functions is provided passing as first parameter a constant string value identifying the quick function; those methods can be used invoking PeaZip from scripts or also creating a link to PeaZip executable with the given first parameter (on any host system).

This allows to have a simple and homogeneous command line interface which masks the complexity and the difference in command line syntax of underlying applications; while through PeaZip's GUI is possible to use underlying applications with great granularity (and save command lines for any further use), PeaZip itself is made accessible through command line to offer an easy access to most common functions.

The full list of strings accepted as quick link to PeaZip functions when passed as first parameter is:

- add2archive: add to a new archive and open PeaZip GUI to set archive's type and options;
- add2pea: add to a new .pea archive;
- add2crypt: add to a new encrypted .pea archive;
- add2split: raw split a single input file;
- add2wipe: securely delete selected file(s);
- add2compare: byte to byte compare two files;
- add27z: add to a new .7z archive;
- add27zmail: add to a new .7z archive and attach it to a mail (requires compatible mail client) \*
- add2separate7z: add each input to a separate new .7z archive;
- add2sfx7z: add to a new self extracting 7z archive (.exe);
- add2sfx7zmail: add to a new self extracting archive and attach it to a mail (requires compatible mail client) \*
- add2zip: add to a new .zip archive;
- add2zipmail: add to a new .zip archive and attach it to a mail (requires compatible mail client) \*
- add2separatezip: add each input to a separate new .zip archive;
- ext2browse: open (and browse if applicable) the archive(s) in PeaZip GUI;
- ext2browsepath: browse the selected folder (or its path, if a file is selected) in PeaZip
- ext2here: extract archive(s) here;
- ext2folder: extract archive(s) here, each in a new folder named after the archive;
- ext2full: extract archive(s), allowing to specify i/o options, password and keyfile;
- ext2to: extract archive(s) to specified folder;
- ext2tofolder: extract archive(s) to specified folder, each in a new folder named after the archive;
- ext2list: list archive(s) content, to quickly look what is in the archive;
- ext2test: test archive(s) content;
- ext2main: extract archives from main applications "Archive extraction" interface;
- ext2commandprompt: open the command prompt in the selected folder (or in its path, if a file is selected)

\*mail functions require a compatible mail client, like i.e. Outlook and Outlook Express, to be the default mail client of the system.

-add2archive, -ext2main and -ext2browse open the PeaZip GUI, to allow further user's interaction.

An example of command line syntax may be: *peazip -add2zip file1 file2 directory3*

which will add specified objects (in the example file 1 and 2, and all content of directory 3) to a .zip archive, auto named after the 1<sup>st</sup> object (in this case will be named file1.zip and will be saved in the same path of file1); using -add27z instead of -add2zip will perform the same task but will result in a .7z archive (-add2pea will result in a .pea archive, -add2sfx7z will result in an self extracting executable and so on).

Another example may be: *peazip -ext2here archive1*

which will extract archive1 in the same path; using -ext2folder archive1 will be extracted to a new folder named "archive1" in the same path of archive1.

# Translations

Lazarus development environment has migrated to UTF-8 for LCL (GUI-related libraries), see UTF-8 support status in [Lazarus](#) and PeaZip documentation, so the application's user interface can now be translated in any language.

Language files are UTF-8 encoded text files which can be edited using any suitable text editor.

To create a new translation file you can:

1. copy en.txt (in PeaZip's path in /res/lang subfolder) or any other language file, if you prefer starting from another language, to a new file;
2. edit lines 2 to 6 of the document to enter language name, PeaZip's **version (major.minor) the translation is aimed to**, translator's and last revisor's name and last revision date;
3. translate the text after the "variable\_name: " part in "=== PeaZip text group ===" AND "=== PeaLauncher text group ===" sections of the file (don't move or remove lines, don't change the "variable\_name: " part);
4. optionally, translate the mini-tutorial after "=== about text group ===" line (free editing, it is loaded and displayed "as is" as application's mini-tutorial).

In "[PeaZip translations](#)" download page, there is a package named peazip-x.y.about\_translations.zip containing a spreadsheet file to help in creating and maintaining localizations, simply compiling column D of the spreadsheet.

The spreadsheet shows variable name (column B), corresponding text string in english (column C) and a blank, yellow column (D) for typing the translated text strings.

On the right, a column E (blue) will show the "variable\_name: " part assembled with the translated string: the content of this area can be copied and paste to replace the text in "=== PeaZip text group ===" and "=== PeaLauncher text group ===" sections (the spreadsheet features TWO pages, one for each of the two groups).

Lines must be pasted in the original order (it is sufficient to sort them by column F).

After column F are featured all currently available translations, in order to help translators more proficient in other languages than English, and to help to spot out what localizations need to be updated.

At each version all language files are mass-updated, with missing text lines in English; to update a localization, it's enough to update the English text lines.

For a better result it is also recommended to check all the language file to see if the update is coherent with linguistic style used by the translator of the current version.

For languages spoken in different ways in different countries (i.e. English, Spanish, Portuguese...) it is recommended to fork the translation, creating i.e. en-us, pt-br etc

PeaZip can load out of order (not optimal for performances) language files for older or newer versions.

**IMPORTANT:** the spreadsheet contains 3 pages, "PeaZip text group", "PeaLauncher text group", and "About text group": all pages need to be completed and pasted (column E, for first two pages) in the language file; the "About text group" can be freely edited.

Translated language files can be sent to [giorgiotani@interfree.it](mailto:giorgiotani@interfree.it) or to the mail address of PeaZip project's page on SourceForge, to be evaluated for inclusion in future updates or publication in "PeaZip translations" packages group.

All translated language files should be considered as released under GFDL, GNU Free Documentation License, as they have to be considered derivate work from the application's language file which is released under [GFDL](#).

Configuration, themes and bookmarks files, saved archive layouts, job logs and exported command lines are saved, for PeaZip 2.2 and beyond, as UTF-8 encoded text.

Older archive layouts (saved as ANSI text by previous versions of PeaZip) can still be used by PeaZip.

It should be noted however that, at present level of development of Lazarus/FreePascal project, most of the underlying FreePascal file-handling routines are still ANSI-only, meaning the UTF-8 strings PeaZip uses internally still have to be translated to ANSI strings to be passed to certain functions.

As PeaZip aims 1) to stay cross-platform and 2) to bridge the gap between GUI and console worlds, allowing to easily export jobs as command lines, UTF-8 support is utterly complicated because each system / desktop environment / widgetset PeaZip is ported to has different levels and ways of supporting UTF-8 encoding for different APIs, for system pipes, for command line interpreters etc.

This issue currently causes:

- PeaZip cannot browse files/dirs containing characters which are not featured in host system's characters set (they will be replaced by "?" wildcard). This is usually not an issue on Linux systems where default character encoding is UTF-8.
- Similarly, archived objects names containing extended characters (over ANSI code 126, ~ character) can be represented as UTF-8 only if the character is featured in host system's console environment characters set. Again, this is usually not an issue on Linux because default encoding is UTF-8. On Windows, currently only single-character encodings and UTF-8 (CP65001) are supported.

Alternatively the extended characters in archived objects names can be replaced by “?” jolly character; this can also be useful to improve command's syntax if the character encoding used in the archive cannot be successfully translated on the current machine.

Anyway, the ability of operate (test, extract etc) on the whole archive is not affected by this issue.

## Notes

### SendTo vs registry

Please note that launching the program through links in SendTo allows receiving multiple input arguments from the command line for each instance of the program; those links can be customized editing the link files in SendTo folder.

Conversely, links in Context Menu can receive a single input argument from command line for each instance of the program; this limitation need changing programming approach to be avoided, see in [this discussion thread](#) in example.

Moreover, Context Menu items can be customized only editing the registry, which is inherently more complex (and potentially more dangerous) than editing a link file as in previous case; that's why links in SendTo was the first system's integration mechanism implemented, anyway Context Menu integration was added in version 1.7 alongside the classic SendTo integration.

At present level of development PeaZip's Context Menu items can accept a single input argument; extraction, test, list, add to separate archives and split feature works just fine with this limitation, since they launch in parallel one instance for each input argument, but add to archive features cannot be replicated (unless there is only a single object to archive).

For that reason it's recommended to keep Add to archive, Add to .7z and Add to zip links in SendTo menu.

For similar reason program's entries in Windows context menu are not grouped in a subfolder since it will involve much more complex (and potentially more dangerous) modifications to be written to registers than creating separate entries.

### Creation of RAR archives

PeaZip can extract, but not create RAR archives.

No free software application can create RAR archives, unless the operation is performed invoking WinRAR binary itself, because the RAR format is proprietary.

Moreover, the UnRAR license explicitly disallows to reverse engineering the file format definitions implemented to build functional RAR creation software.

### ACE archives

Unace may require to interactively respond to some warning messages extracting archives (i.e. if an object with this name jet exists in the output path), or to enter a password for handling an encrypted archive. If the console is not available for user's input (as when calling unace through gwrap executable), the feedback cannot be entered and the application would freeze waiting for feedback.

To avoid this possible stall situation unace extraction and test operations are always called in console mode by PeaZip, while listing operations, as for all other formats, are always launched in graphical mode of operation to offer better output information handling (and since naming conflict and password request conditions cannot be encountered in list mode).

To run test on ACE archive it's recommended to save the command line as text and then execute it as command line or as script, in order to don't get the console's window closed after the operation completes.

Please note that from PeaZip 1.9.1 the program's base packages contain only open source software, so handling ACE archives will require installing a separate plugin (see application's website) as described in “Plugin” chapter. Trying to open an ACE archive without the plugin installed will result in a message to remember the UNACE plugin is required.

### Office, Open Office, MSI and EXE files

Files in those formats are actually containers from different resource objects which can be browsed and extracted by PeaZip; in the archive browser and layout composer those file types are highlighted as supported archive types.

However since those file types are more commonly archived rather than handled as archives themselves, PeaZip by default don't handle them as archives unless they are explicitly opened with “Open archive” function so double clicking on files of those types within PeaZip starts the associated application rather than opening them as archives.

Dragging them on PeaZip will popup a disambiguation message asking if opening them as archives or add them to archive layout interface.

### Compilation, build and porting

PeaZip, Pea and Gwrap are written in FreePascal (highly compatible with Delphi and ObjectPascal languages) and require [Lazarus IDE](#) to be compiled; Windows setup scripts (.iss files) are developed using Inno Setup.

To compile PeaZip binaries open the .lpi file of the desired binary (i.e. peach.lpi for peazip binary) and select “build all”.

FreePascal supports multiple widgetsets (Win32, WinCE, GTK1, GTK2, Qt, Carbon, fpGUI) to allow compilation of GUI applications for the various supported systems and to create different “flavours” of the application for platforms supporting multiple widgetsets (i.e Linux).

PeaZip's sourcecode is cross platform, platform-specific code portions are contained in conditional compilations blocks.

Deploying the application to other targets than MSWINDOWS and LINUX may require adaptation of those platform specific areas (and possibly other fine-tunings); PC-BSD users successfully built PeaZip on \*BSD platform.

PeaZip will also need various backend compression and archiving applications to be reachable in expected directories within the application's path, please refer to the structure of any of the precompiled packages, either installable or portable, to see what third parts binaries must be included, and refer to respective Authors for ports of those utilities.

Being PeaZip programmed as frontend/backend application, missing or unwanted backend binaries can be omitted, at the cost of losing the ability of handling supported formats; for the same reason, backend binaries can be freely replaced with 64 bit counterparts or with updated versions (which will work fine as long as they follow the same syntax).

PeaZip code should be fairly easy to port on Delphi and other Pascal dialects; the underlying [crypto library](#) is explicitly written to be portable to most or all Pascal dialects, however due to some ASM parts some of its features may be x86 processor specific.

From version 2.5 PeaLauncher was extended to be also a standalone extraction application; this could make porting easier at least for basic extraction features, since its user interface is more simpler than PeaZip's one.

### **Qt issues**

Currently, Lazarus/FreePascal IDE support for Qt widgetset, on Linux platform, is marked as beta.

It means under some circumstances some unexpected behaviours may show; know issues are:

- In most cases, the drag and drop from system to the Qt version of the application will not work
- In some distribution Qt PeaZip will miss the close window icon; the application can be closed using File > Quit in main menu (Ctrl+Q keyboard shortcut)