PeaZip

Open source, portable file and archive manager

Document revision: 2009 05 27

Author: Giorgio Tani

Translation: Giorgio Tani

This document refers to:
PeaZip 2.6.1 executable implementation;

PeaZip official site:
http://sourceforge.net/projects/peazip/

Content:

What is PeaZip

PeaZip is a general purpose file and archive manager application.
It aims to provide a cross-platform graphical interface for many Open Source archiving and compression utilities in order to handle most of available archiving formats (Table 1).
The program features powerful and flexible inclusion/exclusion filters and search tools, and allows to deeply fine tune the jobs, exposing through a single, consistent GUI the options of underlying applications.
Once defined, jobs can be exported to command line in order to get the full control on job 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 is 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.

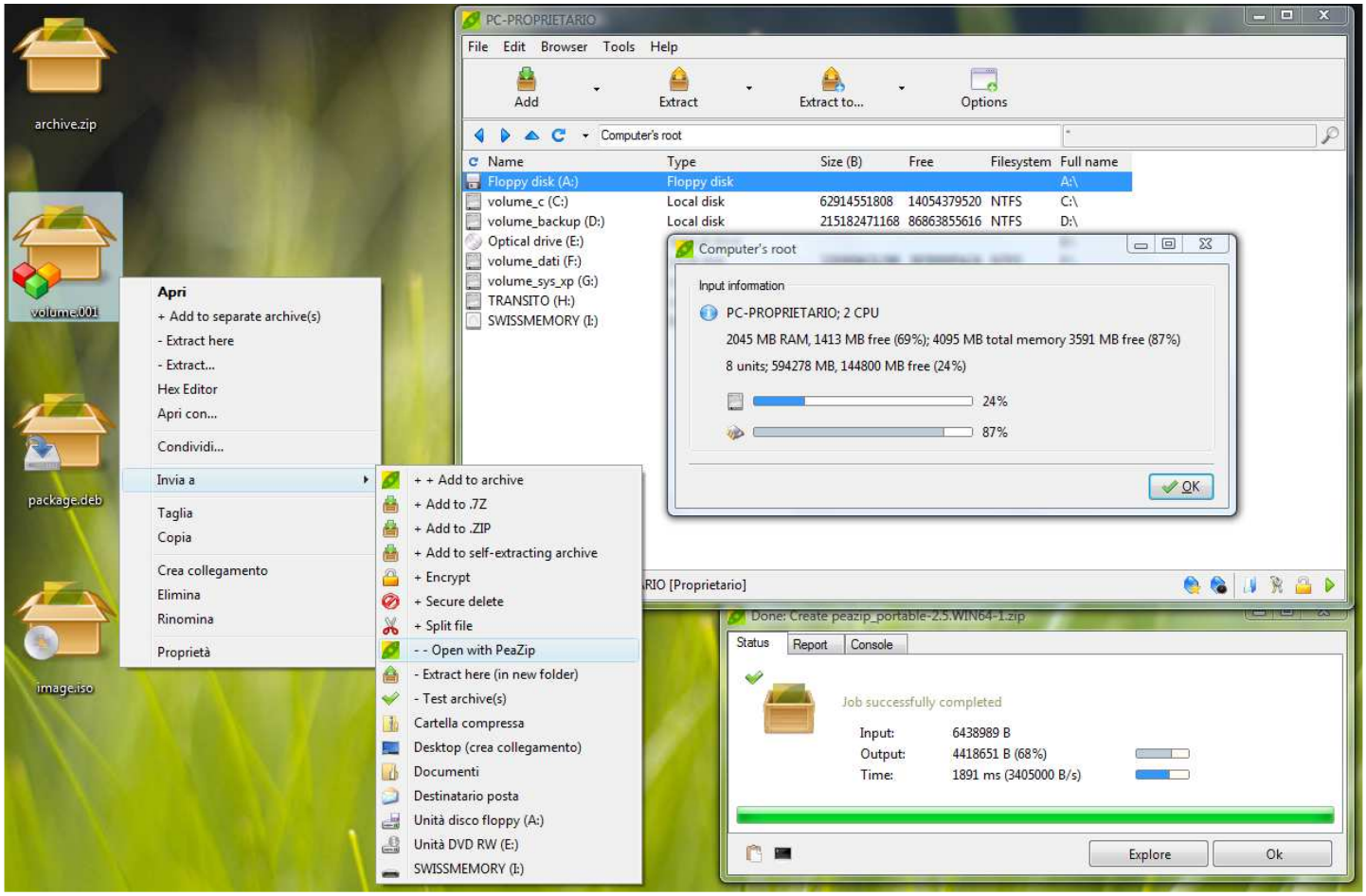


Image 1: high resolution icons, SendTo and context menu integration (PeaZip entries conventionally starts with + and -) and some program's windows in Windows Vista

Packages

The program is available both as installable package and as portable application, not requiring installation (can be used from remote locations, or removable media like USB sticks); both versions shares the same source code and features.

PeaZip (installable) is available as Windows installer and, for Linux platform, as DEB, RPM and TGZ installable packages; the program should be installed once with administrative privileges and then any user starting PeaZip will have their personal configuration and bookmarks files saved in the user's private home (Linux) or application data (Windows) folder.

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. Otherwise you can just rename the package as "setup", since the system will ask the user profile to be used to run the program each time an executable named "setup" is started as unprivileged user.

On uninstall, only the personal settings of the user performing the uninstall operation (likely the administrator) will be removed, other users can decide to remove them manually or to delete configuration or bookmarks (or both) from PeaZip itself when a newer version is available, see “Settings” chapter.
On Windows both SendTo and context menu entries can be customized during setup and most functions can be created as entries either in SendTo and context menu, or in both.

Hint: PeaZip installer for Windows can be used each time it is needed to set program’s file associations and SendTo / context menu entries, it is not needed to uninstall the program before running the same version’s installer again.

PeaZip Portable is available as .zip package for Windows and as .tzr.gz for Linux. To use the program just unpack it to the desired path and start PeaZip executable. PeaZip Portable is available also as PortableApps package.

All program’s packages are linked and explained in the release map.
Sources, documentation, previous versions and all other resources are available on project’s download page.

Plugin

All the open source application for which PeaZip acts as frontend, due to their licensing models, are included in PeaZip pre-built packages for complete ease of use; since release 1.9.1, all PeaZip (and peazip_portable) 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.

Supported file types

See full list on 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
  - ZIP, SMZIP: 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
  - Disk images
    - ISO standard disk image format
    - UDF widespread disk image format
  - Java archives: JAR, EAR, WAR
  - Linux installer formats:
    - DEB (Debian based)
    - PET/PUP (Puppy Linux)
    - RPM (Redhat based)
    - SLP (Stampede Linux)
  - LZMA: single file compressed with LZMA algorithm, introduced with 7z format
  - Macintosh formats: DMG/HFS package/disk image format
  - Microsoft formats
    - CAB compressed archive format
    - Compound
      - MSI Microsoft’s proprietary installer format for Windows
      - Some of MS Office formats: DOC, XLS, PPT
    - CHM, CHW, HXS compressed help files
    - WIM, SWM Windows image format
  - NSIS: Open Source Windows installer 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.
  - PAK, PK3, PK4 : modified zip archives, used to store data by some popular games (like Quake3, Quake4, Doom3)
  - RAR: popular archiving and compression format, with advanced encryption and error recovery features
  - UP3: U3 portable application’s package format

    - XPI: Mozilla package format (addons for Firefox, Thunderbird etc)

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

- Full support
    - PAQ8F/JD/L/O: experimental compressor; at cost of high computing time and memory usage provides best known compression ratio for most data structures (Matt Mahoney); F/L: Matt Mahoney; JD: Bill Pettis; O: Andreas Morphis
    - LPAQ1/5/8: lighter and faster version of PAQ, at the cost of some compression; 1: Matt Mahoney; 5: Alexander Ratushnyak

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 0.40 (GPL)

- Full support
    - FreeArc's ARC/WRC: 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

User interface

“**File**” submenu in main menu contains primary application’s features:
In the first area, **Create archive** activates “layout composer”, the archive creation interface, similar to a CD/DVD-burner application, 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, **Filesystem**, **Bookmarks** and **Recent** submenus represent three alternative ways to quickly access to most used archives or folders in the browser interface, see “File and archive manager” chapter.

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.

“**Tools**” submenu in main menu contains:

- **Run as different user** entry (Windows only), which closes current PeaZip instance and opens a new one with alternative user profile;
- **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;
- **Settings** panel, 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.

On application start-up, PeaZip parses the parameter list elements 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 layout), in order to display the proper interface to the user; see “Customisation and scripting” chapter for the startup parameters that can be passed to PeaZip.
Otherwise, if PeaZip is invoked without additional input parameters (i.e. double clicking on the executable or on a link), the browser interface will be displayed by default; main panel or layout composer can be chosen alternatively, see “Settings” chapter.

File and archive manager

Browser

PeaZip opens by default with file browser interface, pointing to the last visited directory.
Four options tabs (archive format's) Options, (file browser's) Clipboard, Filters, and Console are shown clicking on Options button on the toolbar; see "Options" section of this chapter for detailed description.
To quickly jump to desired locations PeaZip offers a **navigation menu** divided in three sections; this structure is replicated in main menu, address bar button's menu and browser's context menu.

- "**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.

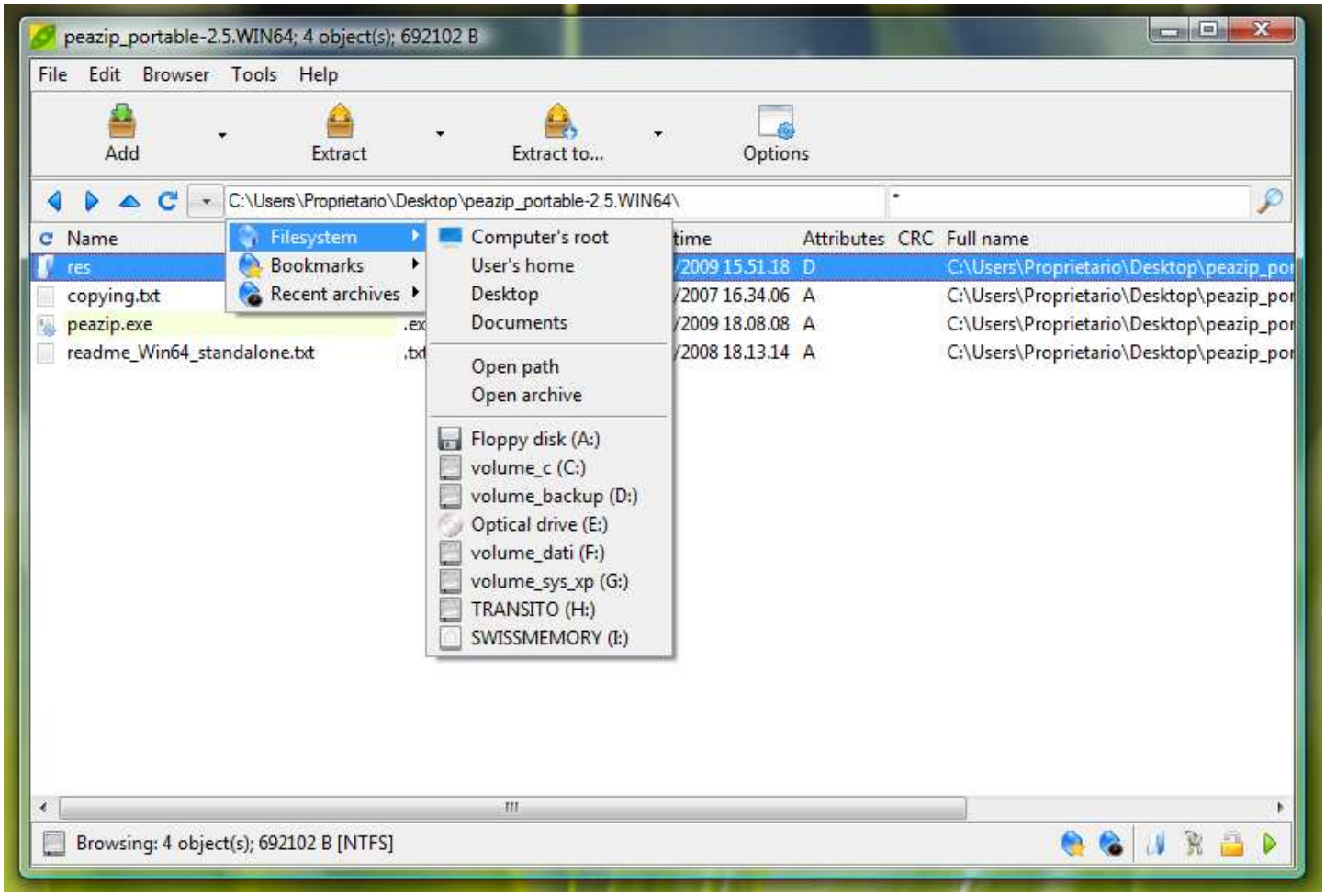


Image 2: browser showing navigation menu

In the browser's **navigation bar** the blue arrows (back, forward, and up) can be used to navigate in previously visited path (or any previously applied filter), as displayed on "History" panel, and to go to upper level.
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 to accept interaction.
Address bar dropdown button can be clicked to display filesystem/bookmarks/recent navigation menu, and the magnifier icon in file/archive browser's navigation bar can be used for basic recursive and non recursive **search** functions; * (string) and ? (single character) wildcards are allowed.
The basic search filters are overridden by advanced filters in "Filters" tab (visible pressing "Options" button in toolbar), for archive types allowing inclusion/exclusion filters (i.e. 7z, rar, tar, zip…).

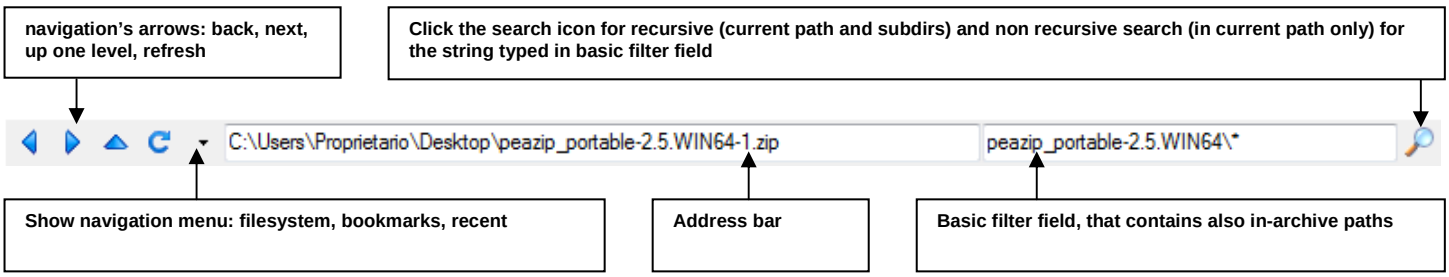


Image 3: navigation bar (Firecrystal theme)

In applications' **toolbar** are featured four buttons.
"**Add**" button, while browsing the filesystem, adds selected objects to the current archive layout, with the button's options menu it is possible to add selected objects to a new empty layout, discarding previous changes, see Create archive chapter.
This button changes in "Add file(s)" while browsing an archive, modifying the current archive; with the button's options menu it is possible to open add directory dialog or launch search dialog to select files and folders to be dragged to the archive.
"**Extract all**" button extracts all selected archives at once, by default to the archive's path or to the last path selected, in current session, with "Extract to" or drag and drop (the destination path is visible as button's hint).
Button's option menu allows to **test** or **list** the archive.
"**Extract all to**" extracts selected items, asking the user for the destination.
To speed up extraction to most commonly used paths, button's options menu allow to select extraction's destination in a menu similar to browser's navigation menu, linking filesystem locations, folders of bookmarked objects and recent archive's paths.
"**Options**" button shows four tabs with advanced program's options, and changes the name in "Browse" until clicked again to return to the browser, see "Options" section in this chapter.

Image 4: toolbar, note, on the right of the first three buttons, the button options dropdown to show additional functions related to the button's main function.

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 the whole 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 functional buttons.
History and bookmarks: "Bookmarks" will show bookmarks panel in the bottom part of the file browser, and "History" will show current session's history panel in the same area; 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 the bookmarks and history panels.
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).

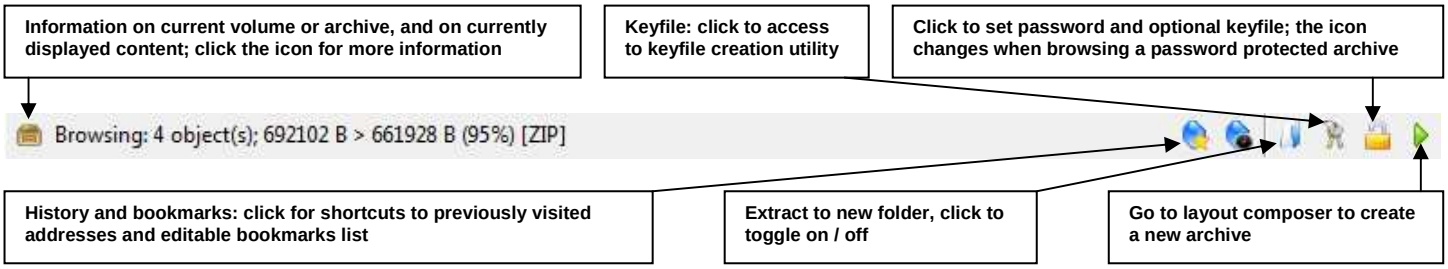


Image 5: status bar

"**Extract in new folder**" button displays and allows to select if the archive(s) will be automatically extracted to a fresh, new folder or not; it is replicated in the context menu. The new folder will have unique name, based on archive's name.
"**Create keyfile**" allows to create a keying file to improve security of encrypted files over password-only encryption (see Keyfiles chapter).

"**Set password and keyfile**" (replicated in context menu) allows to set password and keyfile for browsing/extracting encrypted archives; this information is kept for the current session or until changed, and can be different for each separate instance of the application.
If an archive being browsed is detected as encrypted, the locker icon will change to highlight that, and encrypted content, while browsing in archives, will be shown with a '*' appended to filename.

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.

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.
On the right of the status bar, the green arrow icon allow to jump to the layout composer (see its own chapter) for archive creation.

In the 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 will invert the order.
Folder objects are highlighted in light orange and have the folder icon on the left, such objects can be opened for browsing with double-click.
Archive-type objects are highlighted in light green and can be opened on double-click. If they are contained into and archive they 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 in the same path specified for the output which consequently should have the appropriate access policies desired for the content.

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.

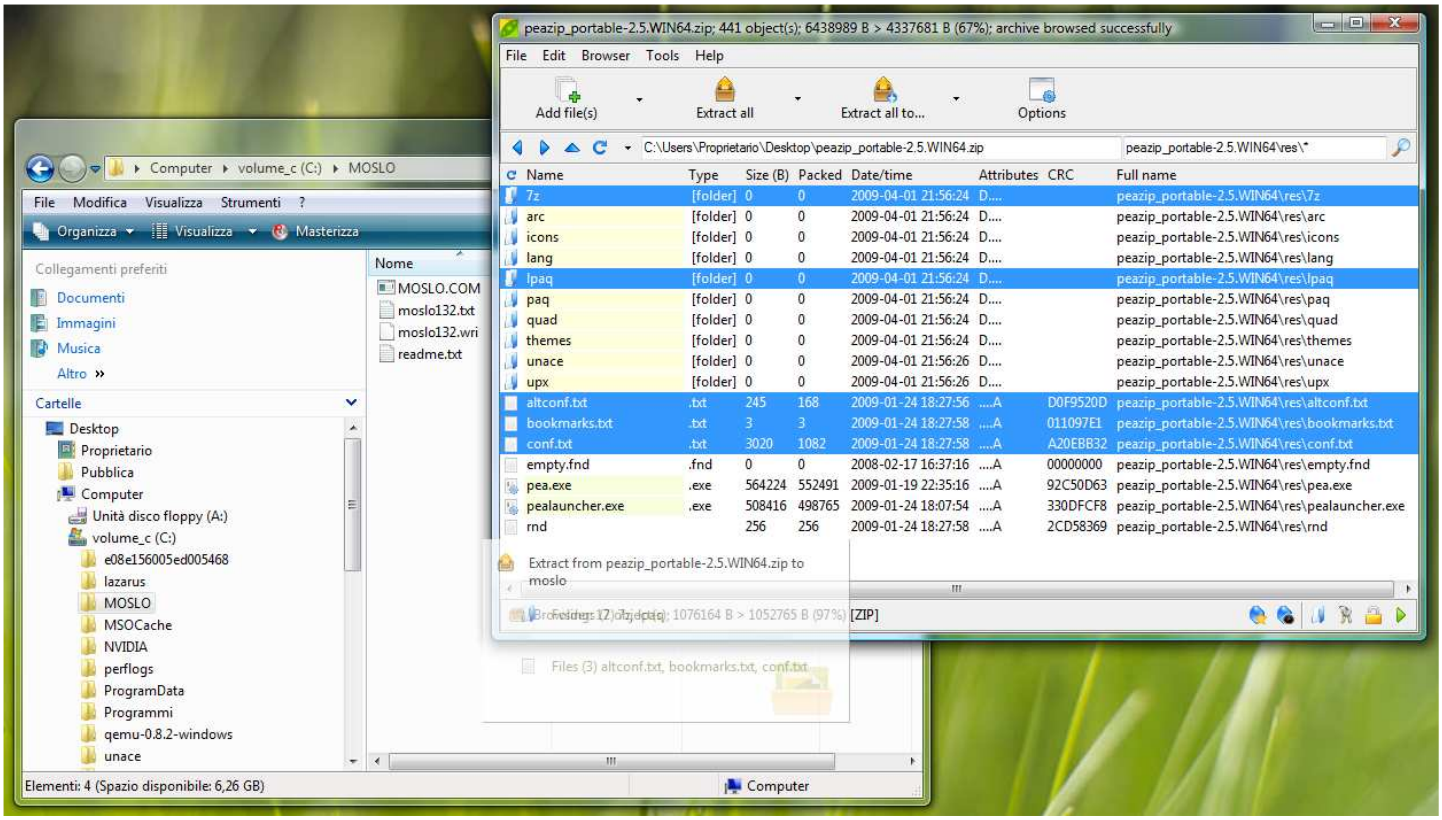


Image 6: drag and drop from application to system (Windows only); files and folders can be dragged from the filesystem or from archives to the desktop or (file)Explorer's windows with Address field enabled (as it is by default on all Windows versions). Drag and drop operations, as well cut/copy selections, are cancelled pressing Esc

The browser can switch between **flat view** mode, displaying all objects contained in the archive, and classic browser mode, using "Toggles browse/flat view" 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 "Tools > Settings > Open archive" it is possible to set PeaZip to start browsing (in 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 supports **drag and drop from system to application**.
Objects dragged to the browser will be added to the archive layout interface, which will be shown allowing to fine tune option before starting archive creation or to return to the browser interface (i.e. in order to add other files).
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.

Browser interface features also a custom **drag and drop to the system** function (in Windows only, see Image 6).
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.

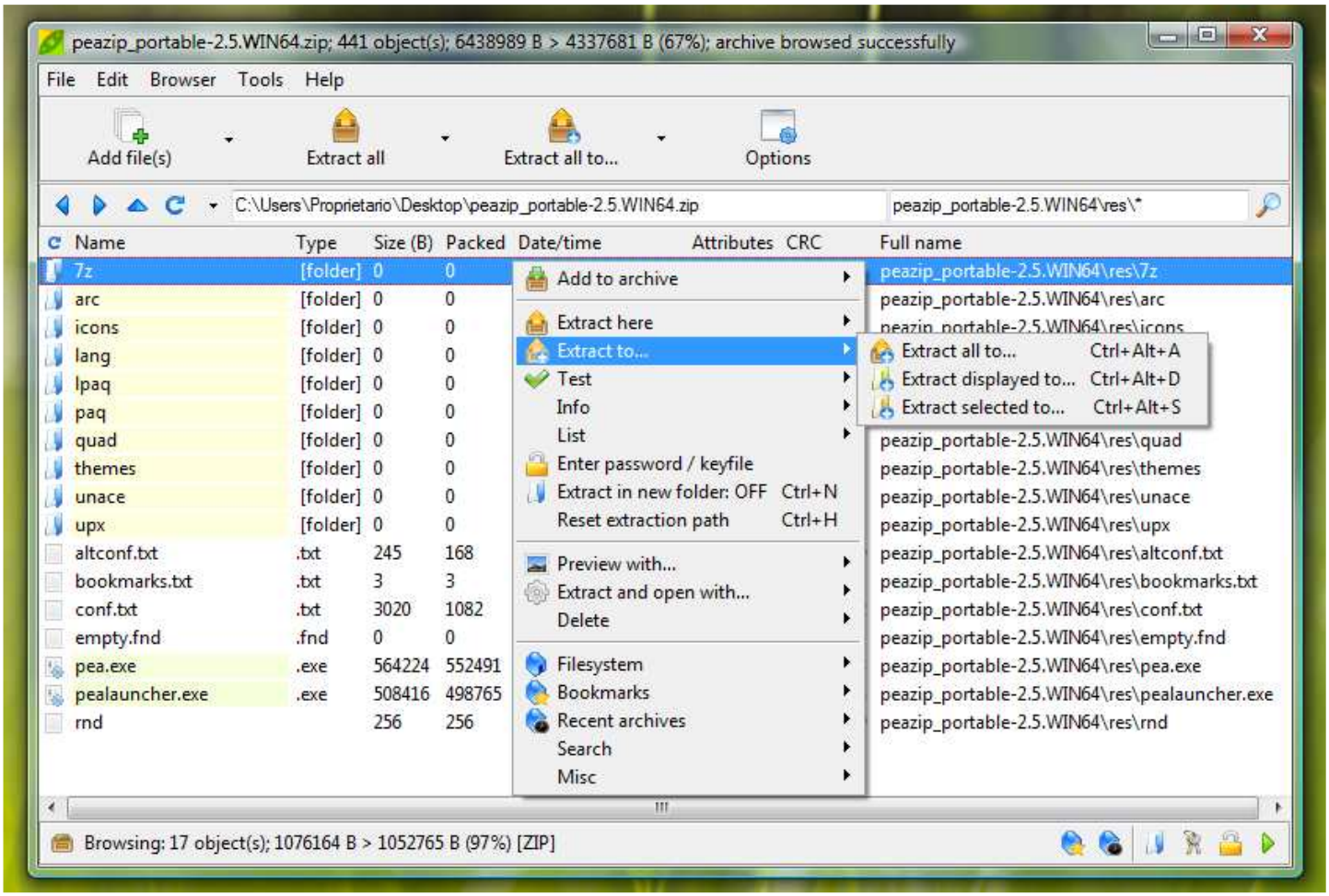


Image 7: the browser, showing context menu.

Note that the locker icon in the context menu and in the status bar is locked when browsing an encrypted archive; 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.

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 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.

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 panel 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.

If same object exists in the archive (same name and same path in archive's directory three), 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

It is alternatively possible to update existing archives also from layout composer interface, using add or update (that replaces existing files only if provided data is newer) instead of "new archive" in function menu in Options tab.

The second area provides **archive management related features**: extract here, extract to, 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.
"Extract all/displayed/selected" group will change caption according to the requested operation in "Options" tab (i.e. if test or list operation are required instead of extraction) and will perform the given operation.
This group's entries will display the output path (by default "here", since default output for extraction is archive's path) and both this group and the "Extract to…" submenu entries will display if extraction to a fresh new directory is programmed, according to input/output options requested in "I/O" tab.
Performing an "Extract to…" function will change the session's default output path (which applies to any extraction function); "Reset extraction path" entry in context menu bring back the output path to the global default (by default, archive's path).
While browsing archives, if it is in a PeaZip's temporary folder "Extract here" functions will automatically extract content outside of the temporary path; otherwise use "Extract to" functions which allows specifying the extraction path.
In this area are featured also "**Set password / keyfile**" and "**Extract to new folder: ON/OFF**" entries previously described.
Please note that while browsing an archive the "Extract all" button, as the "Extract" entries in menu and context menu, act as a jolly as it change action according to the function required in Options tab.

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; in "Tools > Settings > Open archive" tab, the double-click event and "open/preview with associated application" action can be further customized.
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. Cut, copy and paste operations can be more conveniently performed with keyboard shortcuts of Crtl+X, Ctrl+C and Ctrl+V respectively.
"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 (fileystem/bookmarks/recent).
"Search" allows recoursive (search in subpaths too) and non-recoursive 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.

Options

Options button toggle between the browser and the options panel, which contains four tabs

Options

This tab set extraction job parameters; options for less common formats can be separately set in "Special formats" group.

- 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.
  - "Repair" (ARC format only) verifies integrity and tries to repair the archive using the recovery records (if featured in the current archive).
- 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.

Clipboard

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.

Filters

(only using formats supported through 7z binary)
"Filters" tab allows to use multiple inclusion and exclusion criteria, one per line, that can recourse or not subdirs of the archive ("Inclusion/Exclusion filters recourse subdirs" options), bypassing the archive browser's basic filters (address bar is disabled until "Use advanced filters" option is checked); advanced filters will be applied when returning to the browser.
Unchecking the option restores the classic browser mode and brings back to archive's root, if browsing inside an archive.
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.

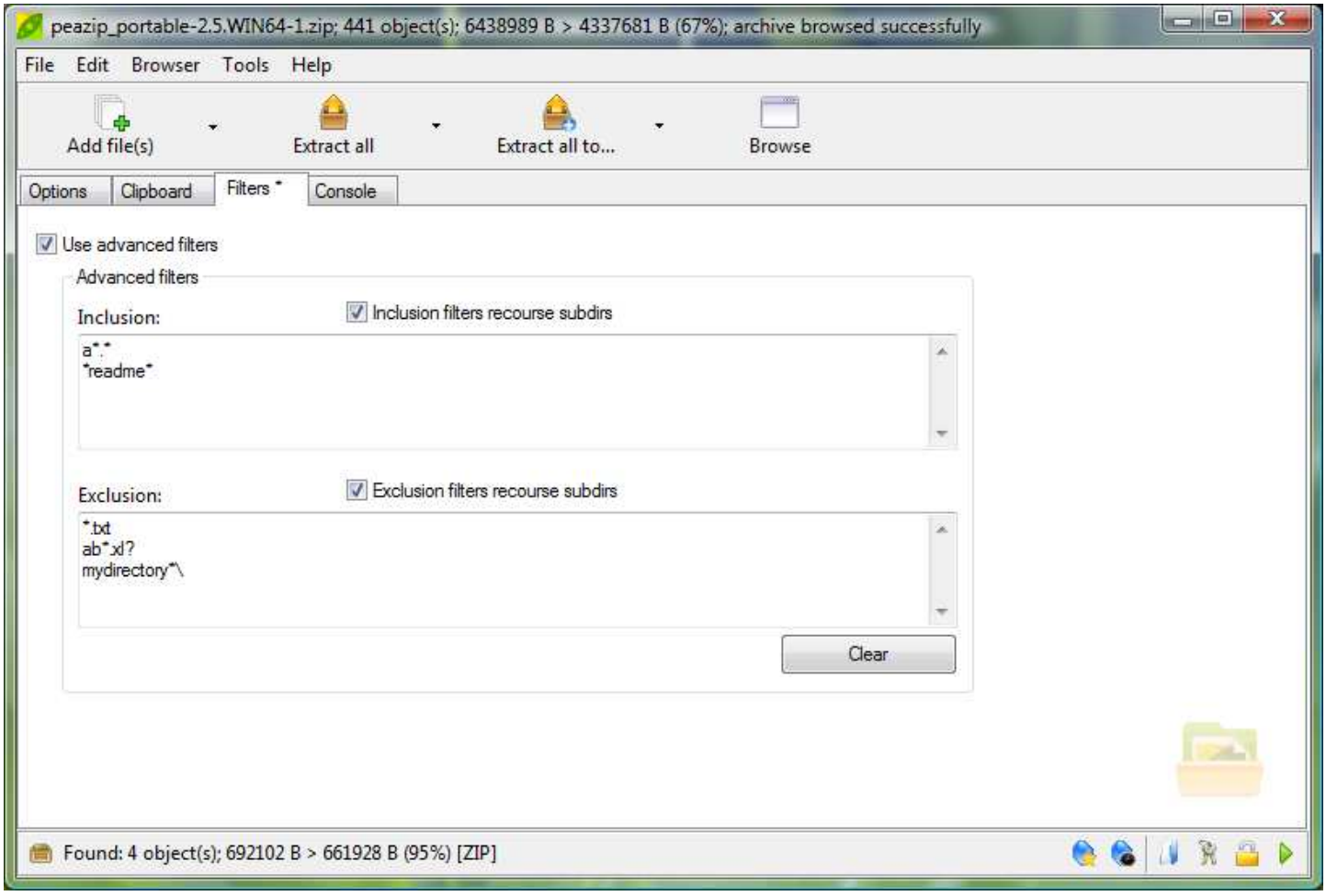


Image 8: inclusion and exclusion filters tab

Console

(only browsing inside archives)

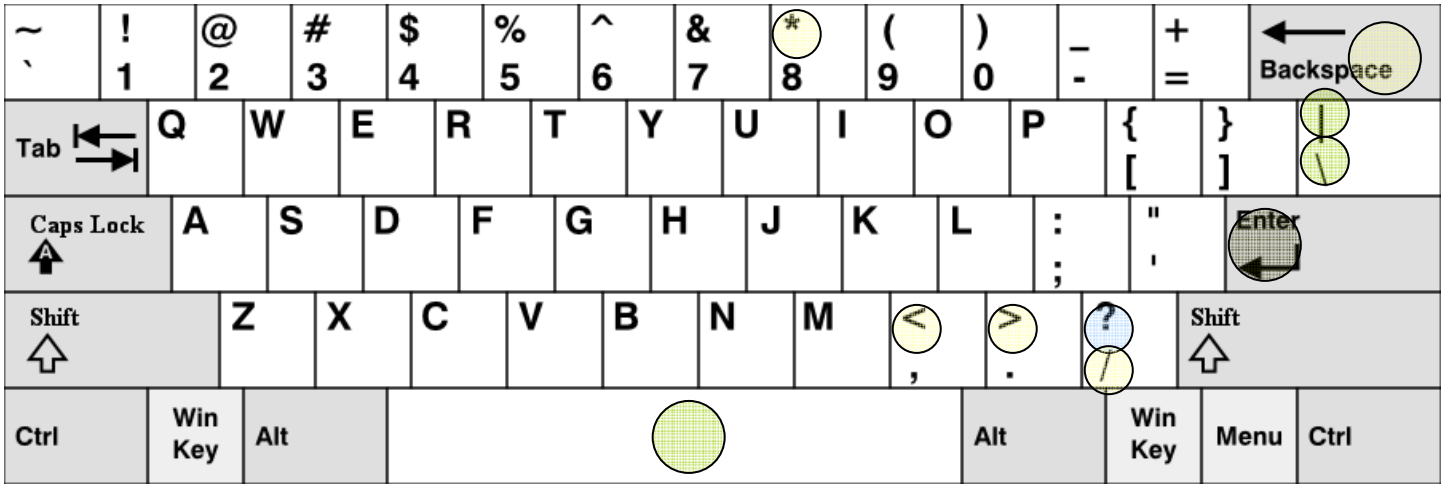
From this tab it is possible to transform the job defined in the GUI interface into a command line that can be edited (independently from the job definition in the GUI frontend); rightclicking on “Click to import…” label it is possible to compose the command line to operate on all, displayed or selected objects (only for file types supporting it, like 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.
Right-clicking on “Save job” button shows a popup menu allowing to chose if applying the job definition to the whole archive, or to selected objects, or to the currently displayed objects only (if partial extraction of content is supported for the current archive type).

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 toggle extract in new folder status / Ctrl+F9 reset extraction path to default / Shift+F9 enter password / Ctrl+Shift+F9 create keyfile
F10 menu
F11 extract selected to
F12 extract selected here



Navigation:

Toggle browse mode **/** flat view mode	* or F6
Go to computer’s or archive’s root	Ctrl+R
Search (in this folder and subfolders)	F3
Search in this folder	Ctrl+F3
Browse most recently visited item	F7
Browse first item in bookmarks list	F8
Open directory/archive	< or Enter or doubleclick on the folder/archive
Up one level	Ctrl+U or > or backspace or click on blue arrow icon or F4
Go to object’s path	Ctrl+P (useful in flat view and search/filter mode)
Go back in history	Ctrl+B or Backspace
Forward in history	Ctrl+F

Extract / extract to:

Extract all content	Ctrl+A
Extract displayed content	Ctrl+D
Extract selected content	Ctrl+S
Extract to… functions	same as previous ones, using Ctrl+Alt+A/D/S

Extract selected here	Ctrl+Enter or | or F12
Extract selected to…	Shift+Enter or Spacebar or F11
Set extract to new folder	Ctrl+N or \ or / or F9
Reset extraction path to default	Ctrl+H or Ctrl+F9

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

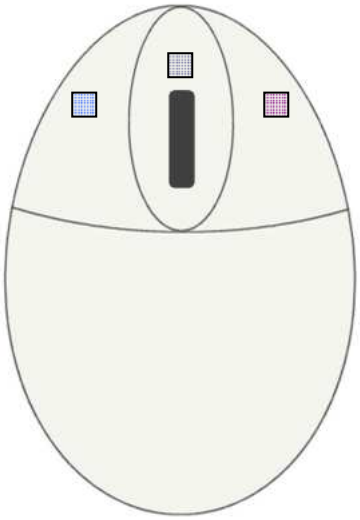
Test:

Test all	Ctrl+T
Test selected objects	?

Modify:

Quick delete / Delete form 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

Mouse controls



Doubleclick: preview selected object with associated application (this action can be customized, see "Settings" chapter);

Rightclick:
on browser area: activate file/archive browser's context menu;
on toolbar buttons and save job button: activate buttons' context menus to apply action to all, displayed or selected objects only;
on home and open path icons: activate quick browse submenu.

Middle button click: extract selected object(s) to…

Keyfiles

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.

PEA natively supports (optional) use of keyfile; for archive formats other than PEA keyfiles can be used since version 2.1.
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 contains an utility to **generate a random keyfile** sampling different entropy sources and submitting entropy collected to a robust random number generation routine; click the key icon in the status bar of the file browser to activate this utility.
In the same interface it's also possible to use the collected entropy 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.
Needless to say, 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.

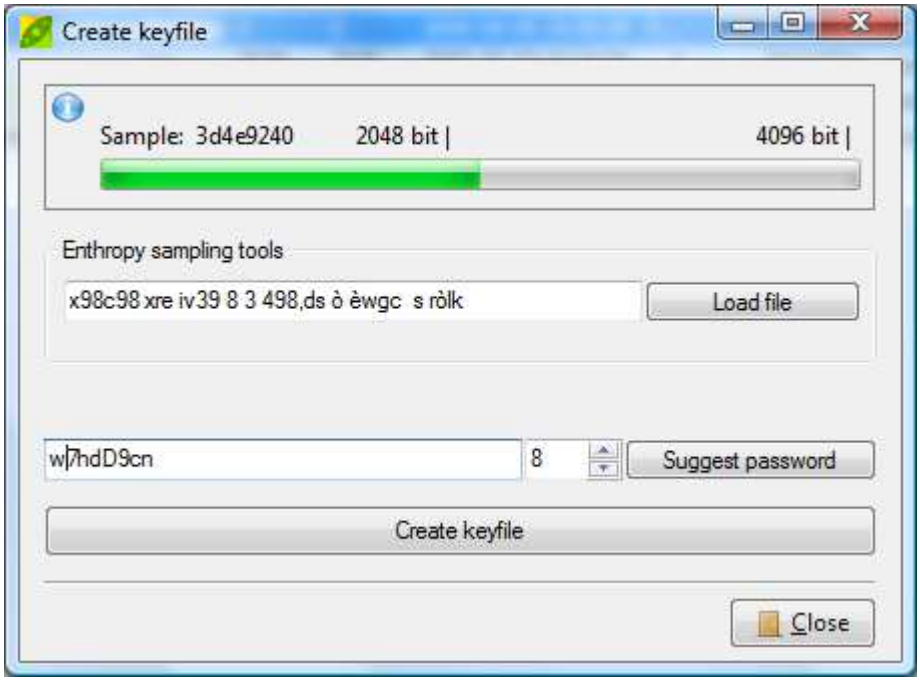


Image 9: keyfile generation utility

File tools

PeaZip collect also handy file management utilities which are non strictly archiving-related.
Those utilities are accessible from both file/archive manager and layout composer 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: 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).

Create archive: layout composer

Layout composer is the interface for creating archives: it offers the user the tools to pack multiple files, directories and even volumes into a single archive that may be split in multiple volumes of a given size.
Four options tabs (archive format's) Options, Input/Output, Filters, and Console are shown clicking on Options button on the toolbar; see "Options" section of this chapter for detailed description.
The layout of the archive, containing the list of objects to be archived, can be saved for future use.
One or more saved layouts can be loaded to populate the current layout (to re-use a saved object selection) or to merge more layouts into a single one, to speed up archiving or backup job definition.

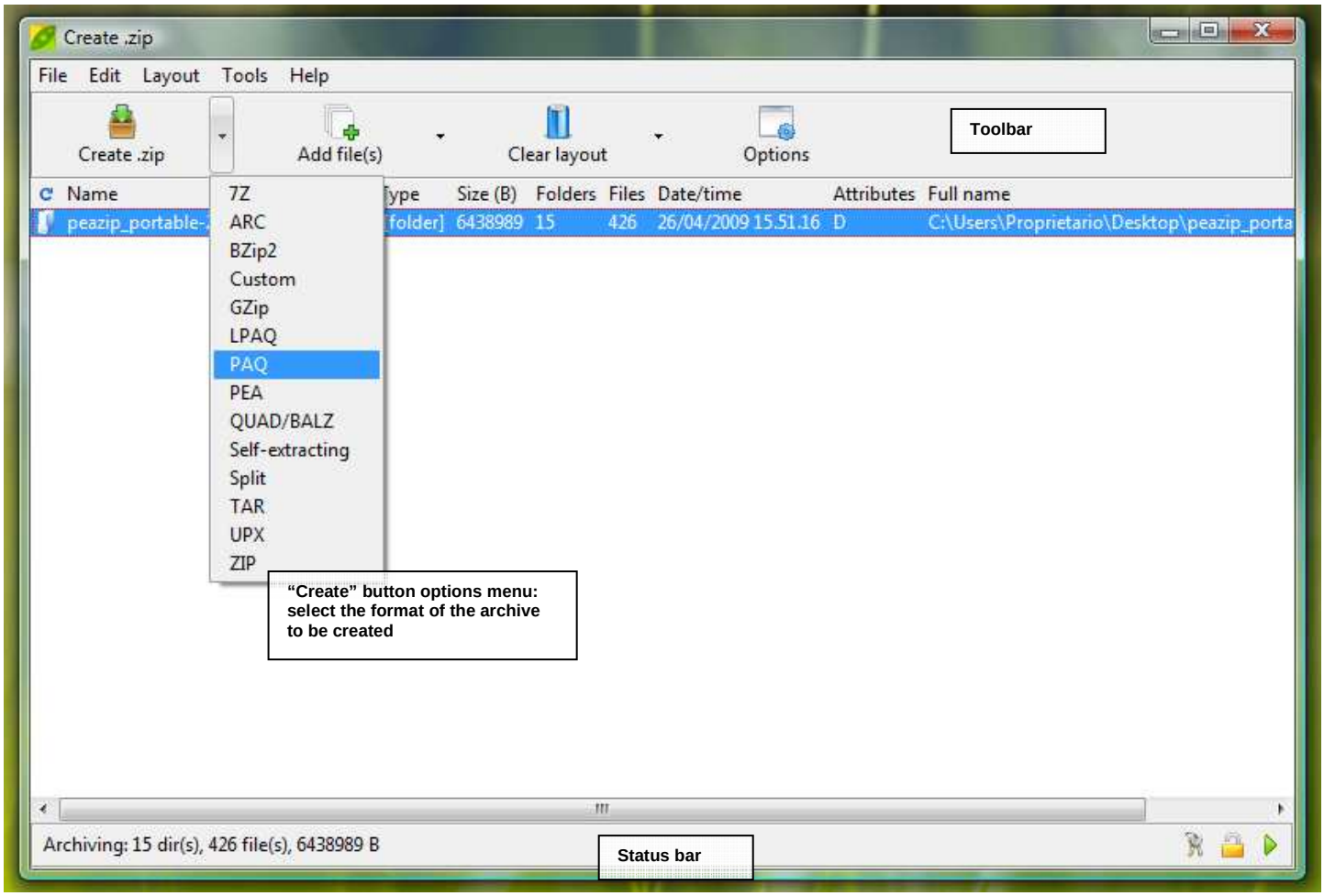


Image 10: "Create archive" interface (on Windows Vista)

Entering the Layout composer interface will enable the "Layout" submenu in main menu, which replicates most of the functions featured on Layout composer's context menu (see below).
Toolbar features four button, first three with button options menus:
Create starts the actual creation of the archive; the button options menu allow to select the archive type, while advanced format options can be reached clicking "Options" button.
Add file(s) open system's file selection dialog; the button options menu allows to add folders (opening system's directory selection dialog) and to launch search dialog to select files and folders to be dragged to the archive layout.
Clear layout remove all items from current layout; the button options menu contains load layout and save layout.
Options button shows four tabs with advanced options, and changes the name in "Layout" until clicked again to return to the archive's layout, see "Options" section in this chapter.

Archive layout composer interface supports **drag and drop** of objects from system to the application: drag items on the layout composer to add them to the current layout; it does not support drag and drop of objects from archive layout to the system, nor cut/copy/paste operations.

*Hint: to create an **encrypted archive** you can set encryption options, as well as password and optionally keyfile, from "Options" 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.

*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.*

The input object list can be populated adding files or folders from the buttons in the application's **toolbar**, or from a **contextual menu**, which is accessible right clicking on the input object's list.
The layout can be saved to a UTF-8 text file (for maximum flexibility of use) and loaded to add a saved selection to current layout; loading layouts it is checked if duplicate objects exists or if non-existing objects are submitted to the current layout (i.e. objects which were moved, renamed, deleted or otherwise made non accessible).
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 object's path.
"Open with…" submenu of context menu allows opening the selected object with PeaZip, associated application, or a custom application.
"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.
In layout composer, folders are highlighted in light orange and have the folder icon on the left; archive types supported by PeaZip are highlighted in light green with archive icon on the left.
The application's **status bar** displays a count of input object's number (files and dirs) and the total size.
On the right of the status bar the green arrow button brings the user to PeaZip's file and archive browser interface, from which more files/folders can be added to the archive's layout.
Key icon and locker icon on the status bar allows, respectively, to open keyfile creation utility and to set encryption (algorithm, password and, optionally, keyfile) for the archive being created.
Please note that selecting a folder as input make all the folder's and subfolder's content to be archived while needing to declare a single object (the folder) in the list; folder's and subfolder's content so will not appear in the list as separate objects but will be counted as size and numbers in the content label; folders in object's list cannot be browsed (native system's dialogs are used to select objects).

Options

Options button toggle between the layout and the options panel, which contains four tabs

Options

This tab contains archive format parameters, like compression level, volume spanning, encryption etc.
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.

"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.

"Volume size" combo box, common to most formats, 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 (in Bytes, KB, MB or GB).
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).
Last used compression options will be remembered to be used as default; resetting applications default (tool button in "default options") resets also compression options.

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

7Z when high compression (with acceptable speed and memory usage, with default settings) 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 those formats, 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.

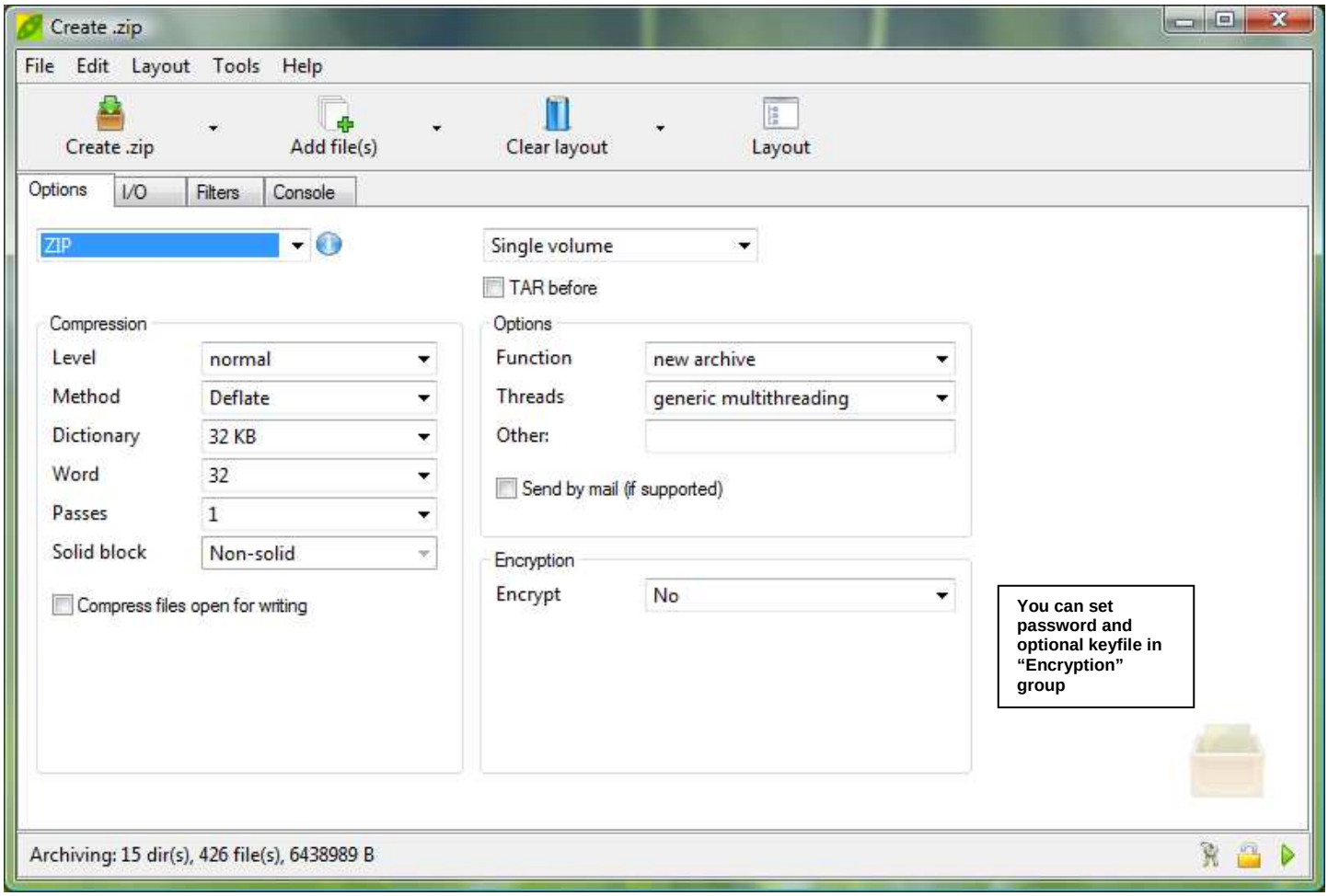


Image 11: archive creation options

"Encryption" group box contains encryption related options:

- "Encrypt" combo box (.7z and .zip only) allow setting a password and (optionally) a keyfile for the archive's content; if "Content and filenames" option is chosen also the archive's content list will be encrypted.
- "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”).

The job definition, as command line to pass to underlying executable, can be saved as UTF-8 text clicking on “Save job” button on the bottom-right corner of the options panel.
7z executable (like other 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 for backend applications is set in Tools > Settings.

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).

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 and it is supported through Pea executable; for more information on it you may look the documentation on the PeaZip’s help page.
Options available for this archive format are:

- compression: three levels of deflate based compression, roughly similar in compression ratio to Zip and GZip, or optionally no compression;
- algorithms for each of the three levels of checks:
  - on input objects: on the uncompressed data structure of input files plus all associated data (object’s name, attributes etc), allowing to detect corruption on original data with object granularity;
  - on stream: in Pea 1.0 file format, implemented in current PEA executable, the stream is the whole archive (storing file’s content after compression and all necessary associated data), allowing to detect corruption on the data structure after compression;
  - on output volume(s): on data in it’s final form (compressed and encrypted, if those features are used), allowing in example to repeat the download of a single corrupted volume;

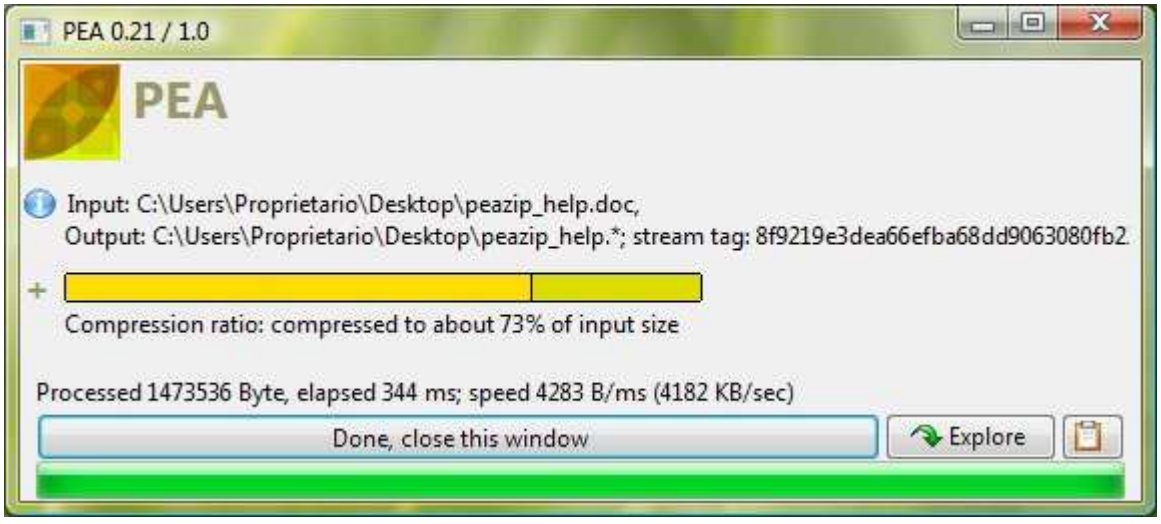


Image 12: Pea, launched by PeaZip; job report is available clicking on “note” icon

Algorithms available for checking objects, stream and volumes ranges from fast checksum like CRC32, to fast hash like MD5, to strong hashes like SHA family.

The stream can be encrypted and authenticated, choosing an Authenticated Encryption instead of an hash or checksum algorithm: the whole content of the archive will be kept private and checked for corruption or tampering in a cryptographically secure way.
Authenticated encryption schemes featured are: 128 bit AES used in classical composition with SHA1 HMAC; 128 bit AES used in EAX mode; 256 bit AES used in EAX mode.
Output archive will be named appending ".000001.pea" to the output name; if a multivolume archive is generated, each volume will bring the progressive number in the file name in place of the .000001 part.
Archiving, compression, volume splitting, redundant integrity check and, optionally, authenticated encryption, are done in a single passage and not requiring temporary files.
Once PEA is launched, the user will be asked for a passphrase (may contain any typeable character) and, optionally, a keyfile if encryption is used, otherwise the archiving operation will start immediately; at the end of the operation PEA allow to access to a detailed job log that can be saved as tabulated text, that can be imported in many kind of application (i.e. spreadsheets or databases) for further inspection.
Pea application will load the same theming parameters used for PeaZip; each Pea instance is a fully independent process and the user can continue using PeaZip for other unrelated tasks.

ARC - *Recommended when high compression (with acceptable speed and memory usage, with default settings) is desired; offers powerful encryption and, optionally, recovery records. Performances scales very well on multicore machines.*
Selecting ARC as archive type, it will be displayed FreeARC's options subpanel, allowing to customize the wide set of options offered by this archiving format.
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).
The format is currently being actively developed.

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)*
Selecting PAQ as archive type, it will be displayed the PAQ options subpanel, allowing to choose compression level and the PAQ8* compressor to be used between supported ones (which uses the same command line syntax); for each compression level the requirement of memory during compression is displayed; please note that if the amount of memory required is higher than available RAM the job will be severely slowed down due to paging to disk.
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.
The positive side in using PAQ is that, for most data structures, it will bring to 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.

LPAQ - *Recommendations are similar to PAQ format, however in comparison LPAQ is fastest and requires less memory*
Selecting LPAQ as archive type, it will be displayed the LPAQ options subpanel, allowing to choose compression level and version of LPAQ compressor to use. LPAQ, likewise GZip and BZip2, is a compression only algorithm; archiving multiple objects with LPAQ compression is possible using "TAR before" switch which consolidate input object in a single TAR archive (temporarily saved in output path) before compression, which is automatically set if needed.

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.

UPX - *Recommended to developers to reduce the size of executables*
Selecting UPX, it will be displayed the Strip/UPX options subpanel, which is intended mainly for developers needing to reduce the size of executables before distribution. This action can accept as input only a single executable file at time and is not intended to be used as a general purpose archiving/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").
It's possible to backup the executable file before Strip/UPX checking "Keep executable's backup" (enabled by default); the backup file will be named as the executable with .backup extension appended.

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.

I/O tab

By default the user will be asked for output path where the archive will be created; by default, PeaZip will remember last used path.
In this tab the user can flag the checkbox "Compress to default path" in order to bypass the request and directly send the compression output to the specified path (clicking on "…" button), which will persist for the current working session; by default, if not specified otherwise, PeaZip will use as output the path of the currently selected object in archive's layout

Hint: default output path can be permanently set in Tools > Settings > Create Archive tab.

The yellow arrow icon resets the output path to the one defined in Settings; the green arrow icon browses the output path.
"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.

Filters

(enabled only when using formats supported through 7z binary)
This tab allows to specify the exclusion of files, directories and file types; please see 7z documentation about exclusion filters to understand how they works.
Saving the archive's layout will save both the object list in "Archive" tab and the filters in this tab; filters are ignored if for the archive format used exclusion filters are not supported (formats not supported through 7z binary).

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).

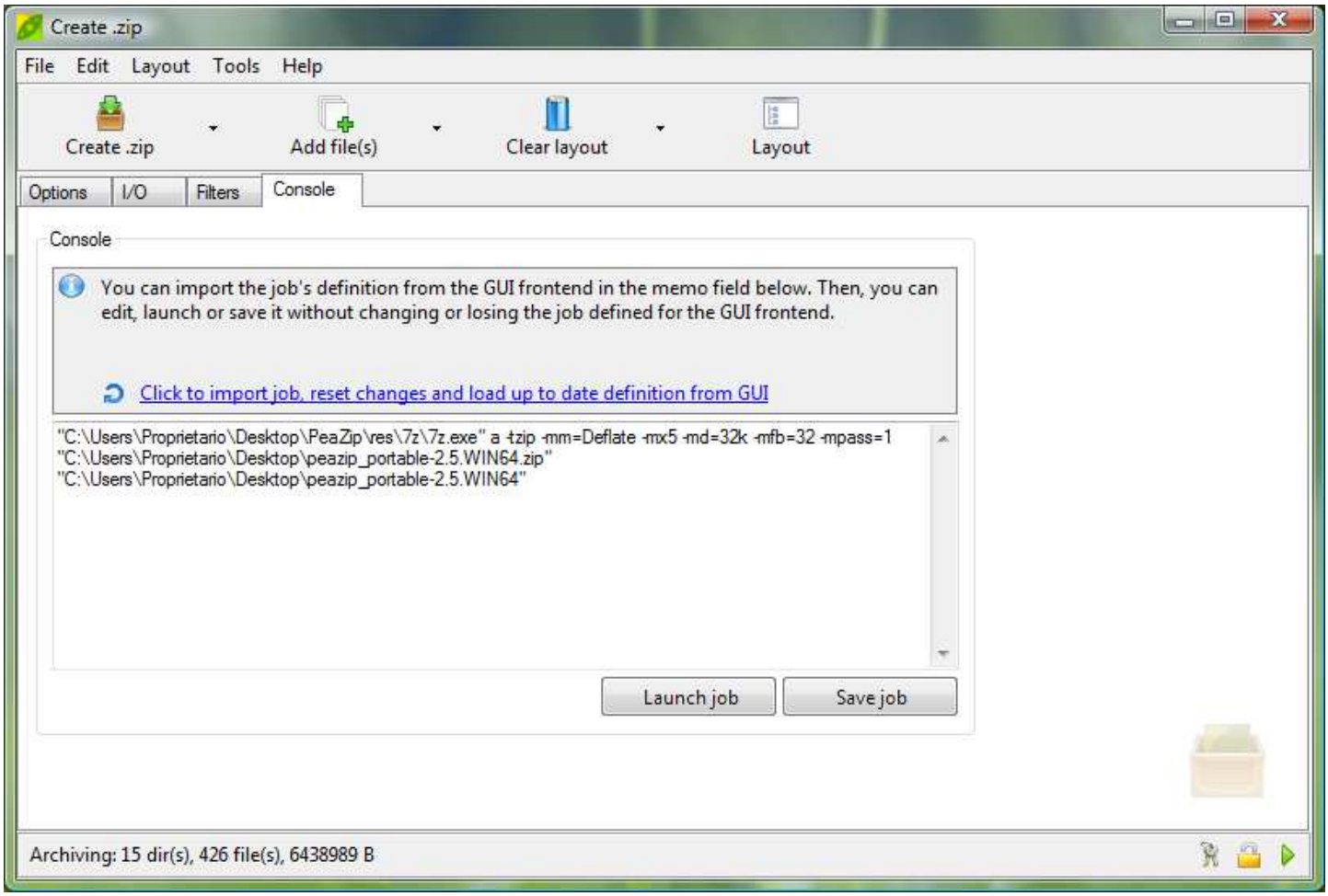
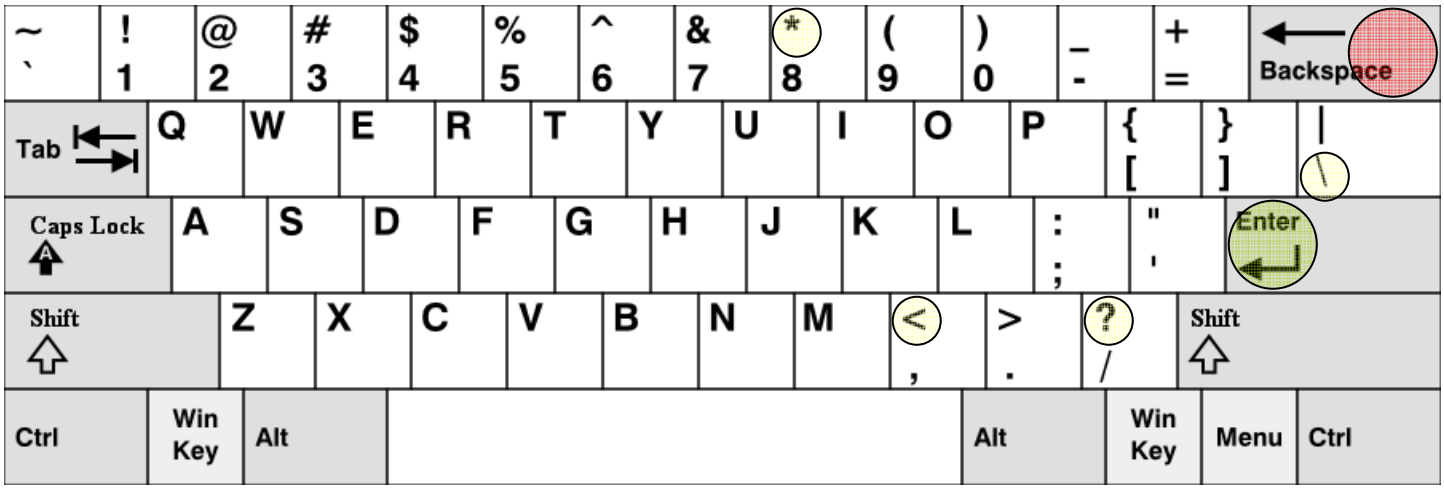


Image 13: console tab

Keyboard shortcuts and other controls

Create archive interface supports following keyboard shortcuts:



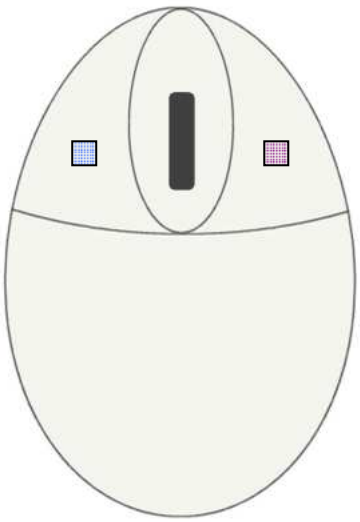
File tools:

Checksum/hash selected file	?
Compared selected file	<
Erase selected files (secure deletion)	* or Shift+Del (Del for quick deletion)
Split selected file	\

Archive layout-related:

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	Ctrl+R or Ctrl+Backspace
Refresh	F5 or refresh icon on the left of layout's titles row

Mouse controls



Doubleclick: open selected object with associated application or browse folder;

Rightclick:
(in archive layout) activate "create layout" context menu;
(on Create button) activate context menu for quick archive format selection, showing only favourite archive formats.

PeaLauncher

PeaLauncher (in /res path) is the job's graphic wrapper, a component that graphically displays an underlying job performed by a console-based application used by PeaZip to create, extract, test or list an archive; it will load the same theming parameters used for PeaZip.
Each PeaLauncher instance is a fully independent process and the user can continue using PeaZip for other unrelated tasks; Pea executable, having its own graphical UI, doesn't rely on 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 detailed job log) even if a different mode is set as default.
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).

If PeaLaucher is invoked with –ext2full option as first parameter (i.e. from "Extract…" menu entries), it allows to set input/output options, password and keyfile before starting the extraction job.
"More options…" link allows to access to PeaZip's archive browser interface for more advanced features (full format-specific options list, partial extraction etc).

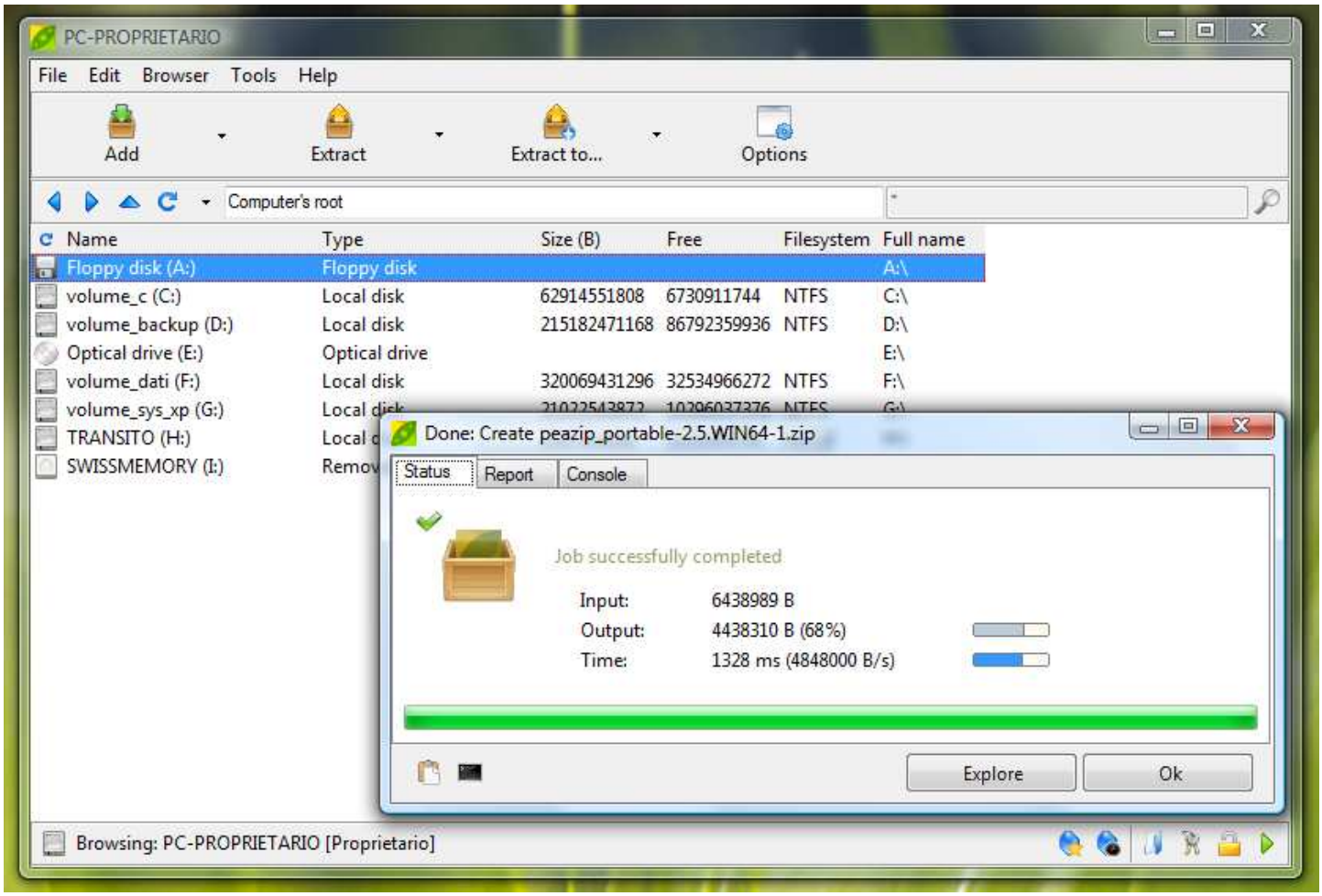


Image 14: PeaLauncher

Until job completition, following control and information elements are shown, as:

- command line that generated the job, displayed in console tab;
- approximate progress indicator;
- if available, it is also reported the progressive amount of output data in bytes and in percentage to input data;
- "Pause" button, which allows to put job in sleep and to resume it when desired;
- "Stop" button terminates the underlying process and will not undo what was done until that moment, leaving the job log and the partial outcome of the job available for further inspection;
- on the lower left corner of the application a graphic shows job's process priority, rightclicking the image displays a popup menu allowing to change job's priority from idle to real time;
- "Close this window when job completes" option is set from Tools > Settings and trigger gwrap window to close after job completition, this behaviour can be overridden for each gwrap instance checking and unchecking the option's checkbox;

- “Halt system when job completes” option can be set in gwrap to make the system shutdown after completition of the present job (useful for long jobs, such backup operations).

After the job completition the user can open the output path (except for list and test operations, which gives no output), save job definition, which is the command line PeaLauncher redirected to the underlying console application (can be useful for learning purpose or to speeding up the creation of scripts), and save the job log.
From version 2.6 PeaLauncher, by default, closes if no errors are detected; it remains open anyway for test, info and list jobs which requires the user to read the job’s report.

To keep the job’s window open after termination, like in previous PeaZip’s version, select “Always keep open” in PeaLauncher combo box in Tools > Settings > General (1).

In console mode, the console will automatically close at job completition without prompting any message if the job completes without errors (otherwise an error report will popup); flagging “Show information messages” in Tools > Settings is possible to change this behaviour, having a confirmation message also when the job completes successfully.

From version 2.5 it 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 application for extraction only.

Settings

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

"**General (1)**" contains application-wide preferences:

- "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.
- "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.
    - "Show password field content", off by default; if enabled allows displaying readable text entered in password fields and disable password confirmation field, allowing the user to write the password only once.
- "Main interface" allows to chose the default interface shown at application's startup: file/archive browser (default) or archive creation, starting with an empty archive layout.
- "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
- "Desktop" allows choosing the path to be used as user's desktop, useful if the application was not able for some reason to identify system's value for users desktop (in this case it will fallback to use user's home) or if the user prefers to use a different path.

"**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.

"**Create archive**" tab:

- "Default format" group allows setting the default archive/compression type; from version 1.2 7z is the default format, but any format with full support (see Table 1) can be selected. 7z is recommended when high compression ratio is needed, Zip is recommended for exchange data with most Windows users while Tar is recommended for exchanging data with UNIX users.
- "Favourite formats" allows choosing a selection of user's favourite formats to be displayed in a popup menu while clicking on create archive button, in layout composer (see Layout composer chapter).
- "Compress to default output path; uncheck to select path each time", unchecked by default: if this switch is off, PeaZip will ask for path each time an archive is created, otherwise it is possible to specify a default path where all new archives will be created without prompting for request to the user; the default path can be reset to a default automatic value which will make PeaZip create new archives in the same folder of the first object listed to be archived, otherwise any reachable path may be specified.

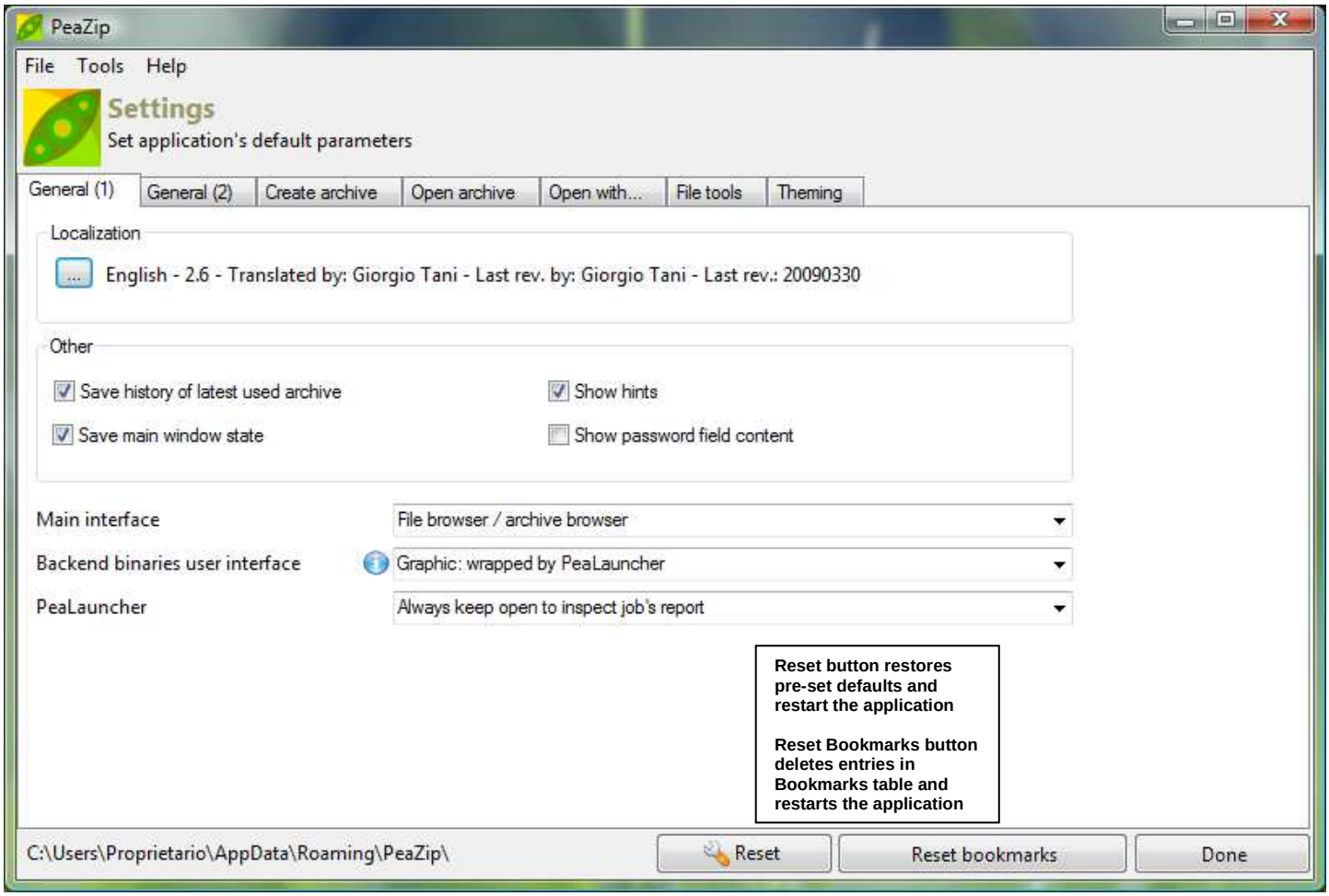


Image 15: default options panels

"**Open archive**" tab:

- "Default extraction path": the path where archives are extracted, by default archive's path itself. When opening archives the user can also specify a different extraction path, which overrides this global default for the current PeaZip's session and which can be changed from "I/O" tab in archive browser interface or each time a different path is specified using an "Extract to" function. The path will be reset to this global default value at each program's startup or using "Reset extraction path" feature in archive browser context menu.
- "Always extract in a new folder": off by default; if you like to be sure to always extract the content into a fresh new directory (named after the archive) flag this option.
- "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
    - "Always ignore paths for Extract displayed" and "Always ignore paths for Extract selected: off by default; ignore paths means extracting the output to output path, not re-creating the directory structure of the output objects 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: extracting a folder from the archive the path will be always preserved, overriding this option for all ignore paths switches.*
    - "Archive browser interface": Browser is a classic navigable browser (default), Flat shows all archive's content, Save last used make PeaZip browsing archives using the last used browser's interface (either Browser or Flat).
    - "Action on open or preview with 'associated application':" set the action to perform on "Extract and open" and "Preview" with associated application (see next point)
        - Extract the object;
        - Extract the object and open the output path;
        - Extract the object and try to open it only with PeaZip, if it's a supported filetype;
        - Extract the object and run it with associated application, tries PeaZip first (default).
        - Extract the object and run it with associated application, not trying PeaZip first.

- "On extract/list/test operations from system's menus" set PeaZip behaviour when an extract, list or test operation (this parameter doesn't apply to add to archive operations) is launched form 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 need, 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.
- "On doubleclick do:" set the action to perform when double-clicking on an object in archive browser (either in flat or browse mode); available actions are
  - "Extract and open with associated application", extracts the object and launch the associated action set at previous point;
  - "Preview with associated application", like the previous, but extracting the object to a temporary path, an hidden folder in output path, which is removed closing the archive (default);
  - "Extract and open with custom application", extracts the object and chose the application to use to run it.

NOTE: if you prefer not relying on temporary files, you can set this option to "Extract and open" (in this way double-click behaviour will be like in 1.8 and previous releases), however please note that temporary objects created by "Preview" function are not created in system's temporary folders, but are rather created in output path, which should logically have the access policy desired by the use for the output.

"**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**" tab:

- "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.

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.

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.

How to…

Using PeaZip (installable) for Windows as reference package, most common operations can be performed as follows.

Archive files and folders:

From PeaZip file browser, select files and folders to be archived and click on “Add” button.
Now you can fine tune the archiving options encryption, compression, volume spanning etc…) in layout composer interface and launch the archive creation with “Create .xxx” button, or return to browser interface with the green arrow in the top right of the application’s toolbar.

From the system, select files and folders to be archived, rightclick and select *“+ + Add to archive” entry in SendTo menu*.

- To directly archive files and folders to a ZIP or 7Z archive, select files, rightclick and select “+ Add to ZIP” or “Add to 7Z” from SendTo menu.

If you have to archive a single object (which may be a file, a directory or even an entire drive) or to **create multiple, separate archives** (one archive for each input item) you can rightclick and select “+ Add to separate archive” from Context Menu.
In this case each input item will placed in a separate archive, i.e. selecting 10 files will result in creating 10 archives, while selecting 2 directories containing 10 files each will result in 2 archives (each directory is passed as a single item).
NOTE: a single input item can be passed to the program in this way (the reason is explained in “Notes” chapter); passing multiple items to “+ Add to separate archive” will result in separate archives created by multiple instances of the program.

- You can use “+ Add to separate *” entries to SendTo or Context Menu to add each selected object to a different, separate, archive in the given * format.
- “+ Add to * and mail” entries in SendTo menu allows to create an archive in * format and send it by mail (requires a compatible mail client, like Outlook, to be the default mail program on the system).

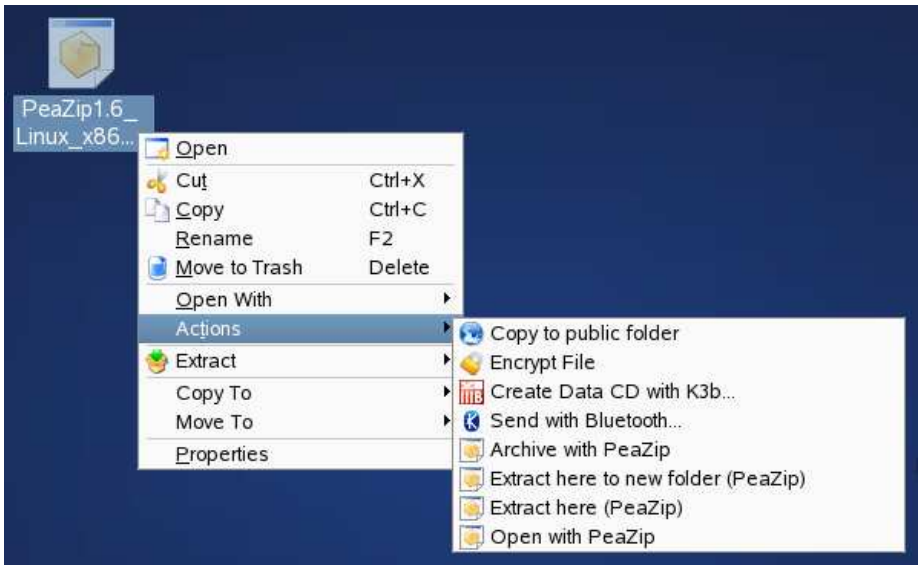


Image 16: menu integration in Linux (KDE, Konqueror Service Menus)

Extract files from archives:

From PeaZip file browser, select one or multiple archives, even of different types, and click on “Extract here” or “Extract to” entries in context menu, or doubleclick on the archive to browse it and use “Extract all” or “Extract all to” buttons in application’s toolbar.
Extract to functions will ask the output directory and change the default path the content will be extract to (by default, the archive’s path).

From the system, select one or more files and use “- Extract archive(s) here” or “- Extract archive(s) to” in Context menu.

To **test archive’s integrity** (if the function is supported by the format) use Test functions in PeaZip or “- Test archive(s)” in SendTo menu; likewise extract and extract to functions it can act on multiple and composite.

From PeaZip’s file browser it is also possible to select files and perform various operations from context menu: **fast deletion** (files are not moved to recycle bin) or **secure deletion**, **split/join file**, **compare files, checksum/hash files,** secure file copy/move with system’s **robocopy** (or xcopy on pre-Vista systems) etc.
Most of those functions are available in layout composer interface context menu as well, and some are available from system’s context menu (like split, encrypt, secure delete).

During installation you can select file types to be associated with PeaZip and functions to be added to SendTo or Context menu (or to both ones); you can run setup whenever you need to reconfigure the program, without need to uninstall it before.
A standalone Windows or Linux package can reach this level of integration as well, and many more internal functions can be integrated using any of the PeaZip's packages, as explained in "Customization and scripting" chapter.
The program parses the first parameter passed to recognize calls to most frequently used functions, so it is possible to call it from scripts, links, .desktop files, registry entries etc depending on the system's available features; any mechanism able to pass parameters to the application's executable is suitable to create calls to internal PeaZip functions.

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-compilant 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;
-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;
-ext2tofolder: extract archives
-ext2list: list archive(s) content, to quickly look what is in the archive;
-ext2test: test archive(s) content

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

-add2archive and -ext2browse open the PeaZip GUI, to allow further user's interaction.
Please note that while from PeaZip interface you may work on a single archive at once, with -ext2browse, -ext2here, -ext2folder, -ext2list and-ext2test you can browse, extract, list and test multiple archives, even of mixed types, at once; operations will be launched in parallel in different instances of PeaZip.

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 1st 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 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 Englis, 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 TWO pages, "PeaZip text group" and "PeaLauncher text group", both pages need to be completed and pasted (column E) in the language file.

Translated language files can be sent to 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 ANSII 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) that 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.