

NAME

unzipsfx - self-extracting stub for prepending to ZIP archives

SYNOPSIS

<name of unzipsfx+archive combo> [-cftuz[ajnoqsCLV\$]] [*file(s)* ... [-x *xfile(s)* ...]]

DESCRIPTION

unzipsfx is a modified version of *unzip(1)* designed to be prepended to existing ZIP archives in order to form self-extracting archives. Instead of taking its first non-flag argument to be the zipfile(s) to be extracted, *unzipsfx* seeks itself under the name by which it was invoked and tests or extracts the contents of the appended archive. Because the executable stub adds bulk to the archive (the whole purpose of which is to be as small as possible), a number of the less-vital capabilities in regular *unzip* have been removed. Among these are the usage (or help) screen, the listing and diagnostic functions (**-l** and **-v**), the ability to decompress older compression formats (the “reduce,” “shrink” and “implode” methods). The ability to extract to a directory other than the current one can be selected as a compile-time option, which is now enabled by default since UnZipSFX version 5.5. Similarly, decryption is supported as a compile-time option but should be avoided unless the attached archive contains encrypted files. Starting with release 5.5, another compile-time option adds a simple “run command after extraction” feature. This feature is currently incompatible with the “extract to different directory” feature and remains disabled by default.

Note that self-extracting archives made with *unzipsfx* are no more (or less) portable across different operating systems than is the *unzip* executable itself. In general a self-extracting archive made on a particular Unix system, for example, will only self-extract under the same flavor of Unix. Regular *unzip* may still be used to extract the embedded archive as with any normal zipfile, although it will generate a harmless warning about extra bytes at the beginning of the zipfile. *Despite this*, however, the self-extracting archive is technically *not* a valid ZIP archive, and PKUNZIP may be unable to test or extract it. This limitation is due to the simplistic manner in which the archive is created; the internal directory structure is not updated to reflect the extra bytes prepended to the original zipfile.

ARGUMENTS

[*file(s)*] An optional list of archive members to be processed. Regular expressions (wildcards) similar to those in Unix *egrep(1)* may be used to match multiple members. These wildcards may contain:

- * matches a sequence of 0 or more characters
- ? matches exactly 1 character
- [...] matches any single character found inside the brackets; ranges are specified by a beginning character, a hyphen, and an ending character. If an exclamation point or a caret (! or ^) follows the left bracket, then the range of characters within the brackets is complemented (that is, anything *except* the characters inside the brackets is considered a match).

(Be sure to quote any character that might otherwise be interpreted or modified by the operating system, particularly under Unix and VMS.)

[-x *xfile(s)*]

An optional list of archive members to be excluded from processing. Since wildcard characters match directory separators (/), this option may be used to exclude any files that are in subdirectories. For example, “fooosfx *. [ch] -x */*” would extract all C source files in the main directory, but none in any subdirectories. Without the **-x** option, all C source files in all directories within the zipfile would be extracted.

If *unzipsfx* is compiled with SFX_EXDIR defined, the following option is also enabled:

[-d *exdir*]

An optional directory to which to extract files. By default, all files and subdirectories are recreated in the current directory; the **-d** option allows extraction in an arbitrary directory (always assuming one has permission to write to the directory). The option and directory may be concatenated without any white space between them, but note that this may cause normal shell behavior to be suppressed. In particular, “-d ~” (tilde) is expanded by Unix C shells into the name of the user’s home directory, but “-d ~” is treated as a literal subdirectory “~” of the current directory.

OPTIONS

unzipsfx supports the following *unzip(1)* options: **-c** and **-p** (extract to standard output/screen), **-f** and **-u** (freshen and update existing files upon extraction), **-t** (test archive) and **-z** (print archive comment). All normal listing options (**-l**, **-v** and **-Z**) have been removed, but the testing option (**-t**) may be used as a “poor man’s” listing. Alternatively, those creating self-extracting archives may wish to include a short listing in the zipfile comment.

See *unzip(1)* for a more complete description of these options.

MODIFIERS

unzipsfx currently supports all *unzip(1)* modifiers: **-a** (convert text files), **-n** (never overwrite), **-o** (overwrite without prompting), **-q** (operate quietly), **-C** (match names case-insensitively), **-L** (convert uppercase-OS names to lowercase), **-j** (junk paths) and **-V** (retain version numbers); plus the following operating-system specific options: **-X** (restore VMS owner/protection info), **-s** (convert spaces in filenames to underscores [DOS, OS/2, NT]) and **-\$** (restore volume label [DOS, OS/2, NT, Amiga]).

(Support for regular ASCII text-conversion may be removed in future versions, since it is simple enough for the archive’s creator to ensure that text files have the appropriate format for the local OS. EBCDIC conversion will of course continue to be supported since the zipfile format implies ASCII storage of text files.)

See *unzip(1)* for a more complete description of these modifiers.

ENVIRONMENT OPTIONS

unzipsfx uses the same environment variables as *unzip(1)* does, although this is likely to be an issue only for the person creating and testing the self-extracting archive. See *unzip(1)* for details.

DECRYPTION

Decryption is supported exactly as in *unzip(1)*; that is, interactively with a non-echoing prompt for the password(s). See *unzip(1)* for details. Once again, note that if the archive has no encrypted files there is no reason to use a version of *unzipsfx* with decryption support; that only adds to the size of the archive.

AUTORUN COMMAND

When *unzipsfx* was compiled with `CHEAP_SFX_AUTORUN` defined, a simple “command autorun” feature is supported. You may enter a command into the Zip archive comment, using the following format:

```
$AUTORUN$>[command line string]
```

When *unzipsfx* recognizes the “\$AUTORUN\$>” token at the beginning of the Zip archive comment, the remainder of the first line of the comment (until the first newline character) is passed as a shell command to the operating system using the C rtl “system” function. Before executing the command, *unzipsfx* displays the command on the console and prompts the user for confirmation. When the user has switched off prompting by specifying the **-q** option, autorun commands are never executed.

In case the archive comment contains additional lines of text, the remainder of the archive comment following the first line is displayed normally, unless quiet operation was requested by supplying a **-q** option.

EXAMPLES

To create a self-extracting archive *letters* from a regular zipfile *letters.zip* and change the new archive’s permissions to be world-executable under Unix:

```
cat unzipsfx letters.zip > letters
chmod 755 letters
zip -A letters
```

To create the same archive under MS-DOS, OS/2 or NT (note the use of the **/b** [binary] option to the *copy* command):

```
copy /b unzipsfx.exe+letters.zip letters.exe
zip -A letters.exe
```

Under VMS:

```
copy unzipsfx.exe,letters.zip letters.exe
letters == "$currentdisk:[currentdir]letters.exe"
```

```
zip -A letters.exe
```

(The VMS *append* command may also be used. The second command installs the new program as a “foreign command” capable of taking arguments. The third line assumes that Zip is already installed as a foreign command.) Under AmigaDOS:

```
MakeSFX letters letters.zip UnZipSFX
```

(MakeSFX is included with the UnZip source distribution and with Amiga binary distributions. “zip -A” doesn’t work on Amiga self-extracting archives.) To test (or list) the newly created self-extracting archive:

```
letters -t
```

To test *letters* quietly, printing only a summary message indicating whether the archive is OK or not:

```
letters -tqq
```

To extract the complete contents into the current directory, recreating all files and subdirectories as necessary:

```
letters
```

To extract all *.txt files (in Unix quote the “*“):

```
letters *.txt
```

To extract everything *except* the *.txt files:

```
letters -x *.txt
```

To extract only the README file to standard output (the screen):

```
letters -c README
```

To print only the zipfile comment:

```
letters -z
```

LIMITATIONS

The principle and fundamental limitation of *unzipsfx* is that it is not portable across architectures or operating systems, and therefore neither are the resulting archives. For some architectures there is limited portability, however (e.g., between some flavors of Intel-based Unix).

Another problem with the current implementation is that any archive with “junk” prepended to the beginning technically is no longer a zipfile (unless *zip(1)* is used to adjust the zipfile offsets appropriately, as noted above). *unzip(1)* takes note of the prepended bytes and ignores them since some file-transfer protocols, notably MacBinary, are also known to prepend junk. But PKWARE’s archiver suite may not be able to deal with the modified archive unless its offsets have been adjusted.

unzipsfx has no knowledge of the user’s PATH, so in general an archive must either be in the current directory when it is invoked, or else a full or relative path must be given. If a user attempts to extract the archive from a directory in the PATH other than the current one, *unzipsfx* will print a warning to the effect, “can’t find myself.” This is always true under Unix and may be true in some cases under MS-DOS, depending on the compiler used (Microsoft C fully qualifies the program name, but other compilers may not). Under OS/2 and NT there are operating-system calls available that provide the full path name, so the archive may be invoked from anywhere in the user’s path. The situation is not known for AmigaDOS, Atari TOS, MacOS, etc.

As noted above, a number of the normal *unzip(1)* functions have been removed in order to make *unzipsfx* smaller: usage and diagnostic info, listing functions and extraction to other directories. Also, only stored and deflated files are supported. The latter limitation is mainly relevant to those who create SFX archives, however.

VMS users must know how to set up self-extracting archives as foreign commands in order to use any of *unzipsfx*’s options. This is not necessary for simple extraction, but the command to do so then becomes, e.g., “run letters” (to continue the examples given above).

unzipsfx on the Amiga requires the use of a special program, MakeSFX, in order to create working self-

extracting archives; simple concatenation does not work. (For technically oriented users, the attached archive is defined as a “debug hunk.”) There may be compatibility problems between the ROM levels of older Amigas and newer ones.

All current bugs in *unzip(1)* exist in *unzipsfx* as well.

DIAGNOSTICS

unzipsfx's exit status (error level) is identical to that of *unzip(1)*; see the corresponding man page.

SEE ALSO

funzip(1), *unzip(1)*, *zip(1)*, *zipcloak(1)*, *zipgrep(1)*, *zipinfo(1)*, *zipnote(1)*, *zipsplit(1)*

URL

The Info-ZIP home page is currently at

<http://www.info-zip.org/pub/infozip/>

or

<ftp://ftp.info-zip.org/pub/infozip/> .

AUTHORS

Greg Roelofs was responsible for the basic modifications to UnZip necessary to create UnZipSFX. See *unzip(1)* for the current list of Zip-Bugs authors, or the file CONTRIBS in the UnZip source distribution for the full list of Info-ZIP contributors.