NAME

unzip − list, test and extract compressed files in a ZIP archive

SYNOPSIS

unzip [**−Z**] [**−cflptTuvz**[**abjnoqsCDKLMUVWX$/:^**]] *file*[*.zip*] [*file(s) . . .*] [**−x** *xfile(s) . . .*] [**−d** *exdir*]

DESCRIPTION

unzip will list, test, or extract files from a ZIP archive, commonly found on MS-DOS systems. The default behavior (with no options) is to extract into the current directory (and subdirectories below it) all files from the specified ZIP archive. A companion program,*zip*(1), creates ZIP archi ves; both programs are compatible with archives created by PKWARE's *PKZIP* and *PKUNZIP* for MS-DOS, but in many cases the program options or default behaviors differ.

ARGUMENTS

file[*.zip*]

Path of the ZIP archive(s). If the file specification is a wildcard, each matching file is processed in an order determined by the operating system (or file system). Only the filename can be a wildcard; the path itself cannot. Wildcard expressions are similar to those supported in commonly used Unix shells (*sh*, *ksh*, *csh*) and 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). To specify a verbatim left bracket, the three-character sequence "[[]" has to be used.

(Be sure to quote any character that might otherwise be interpreted or modified by the operating system, particularly under Unix and VMS.) If no matches are found, the specification is assumed to be a literal filename; and if that also fails, the suffix `.zip` is appended. Note that self-extracting ZIP files are supported, as with any other ZIP archive; just specify the `.exe` suffix (if any) explicitly.

[*file(s)*] An optional list of archive members to be processed, separated by spaces. (VMS versions compiled with VMSCLI defined must delimit files with commas instead. See **−v** in **OPTIONS** below.) Regular expressions (wildcards) may be used to match multiple members; see above. Again, be sure to quote expressions that would otherwise be expanded or modified by the operating system.

[**−x** *xfile(s)*]

An optional list of archive members to be excluded from processing. Since wildcard characters normally match ('/') directory separators (for exceptions see the option **−W**), this option may be used to exclude any files that are in subdirectories. For example, "`unzip foo *.[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.

[**−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). This option need not appear at the end of the command line; it is also accepted before the zipfile specification (with the normal options), immediately after the zipfile specification, or between the *file(s)* and the **−x** option. 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

Note that, in order to support obsolescent hardware, *unzip*'s usage screen is limited to 22 or 23 lines and should therefore be considered only a reminder of the basic *unzip* syntax rather than an exhaustive list of all possible flags. The exhaustive list follows:

−Z *zipinfo*(1) mode. If the first option on the command line is **−Z**, the remaining options are taken to be *zipinfo*(1) options. See the appropriate manual page for a description of these options.

−A [OS/2, Unix DLL] print extended help for the DLL's programming interface (API).

−c extract files to stdout/screen (''CRT''). This option is similar to the **−p** option e xcept that the name of each file is printed as it is extracted, the **−a** option is allowed, and ASCII-EBCDIC conversion is automatically performed if appropriate. This option is not listed in the *unzip* usage screen.

−f freshen existing files, i.e., extract only those files that already exist on disk and that are newer than the disk copies. By default *unzip* queries before overwriting, but the **−o** option may be used to suppress the queries. Note that under many operating systems, the TZ (timezone) environment variable must be set correctly in order for **−f** and **−u** to work properly (under Unix the variable is usually set automatically). The reasons for this are somewhat subtle but have to do with the differences between DOS-format file times (always local time) and Unix-format times (always in GMT/UTC) and the necessity to compare the two. A typical TZ value is ''PST8PDT'' (US Pacific time with automatic adjustment for Daylight Savings Time or ''summer time'').

−l list archive files (short format). The names, uncompressed file sizes and modification dates and times of the specified files are printed, along with totals for all files specified. If UnZip was compiled with OS2_EAS defined, the **−l** option also lists columns for the sizes of stored OS/2 extended attributes (EAs) and OS/2 access control lists (ACLs). In addition, the zipfile comment and individual file comments (if any) are displayed. If a file was archived from a single-case file system (for example, the old MS-DOS FAT file system) and the **−L** option was given, the filename is converted to lowercase and is prefixed with a caret (^).

−p extract files to pipe (stdout). Nothing but the file data is sent to stdout, and the files are always extracted in binary format, just as they are stored (no conversions).

−t test archive files. This option extracts each specified file in memory and compares the CRC (cyclic redundancy check, an enhanced checksum) of the expanded file with the original file's stored CRC value.

−T [most OSes] set the timestamp on the archive(s) to that of the newest file in each one. This corresponds to *zip*'s **−go** option except that it can be used on wildcard zipfiles (e.g., ''`unzip −T \*.zip`'') and is much faster.

−u update existing files and create new ones if needed. This option performs the same function as the **−f** option, extracting (with query) files that are newer than those with the same name on disk, and in addition it extracts those files that do not already exist on disk. See **−f** above for information on setting the timezone properly.

−v list archive files (verbose format) or show diagnostic version info. This option has evolved and now behaves as both an option and a modifier. As an option it has two purposes: when a zipfile is specified with no other options, **−v** lists archive files verbosely, adding to the basic **−l** info the compression method, compressed size, compression ratio and 32-bit CRC. In contrast to most of the competing utilities, *unzip* removes the 12 additional header bytes of encrypted entries from the compressed size numbers. Therefore, compressed size and compression ratio figures are independent of the entry's encryption status and show the correct compression performance. (The complete size of the encrypted compressed data stream for zipfile entries is reported by the more verbose *zipinfo*(1) reports, see the separate manual.) When no zipfile is specified (that is, the complete command is simply ''`unzip −v`''), a diagnostic screen is printed. In addition to the normal header with release date and version, *unzip* lists the home Info-ZIP ftp site and where to find a list of other ftp and non-ftp sites; the target operating system for which it was compiled, as well as

(possibly) the hardware on which it was compiled, the compiler and version used, and the compilation date; any special compilation options that might affect the program's operation (see also **DECRYPTION** below); and any options stored in environment variables that might do the same (see **ENVIRONMENT OPTIONS** below). As a modifier it works in conjunction with other options (e.g., **−t**) to produce more verbose or debugging output; this is not yet fully implemented but will be in future releases.

−z display only the archive comment.

MODIFIERS

−a convert text files. Ordinarily all files are extracted exactly as they are stored (as ''binary'' files). The **−a** option causes files identified by *zip* as text files (those with the 't' label in *zipinfo* listings, rather than 'b') to be automatically extracted as such, converting line endings, end-of-file characters and the character set itself as necessary. (For example, Unix files use line feeds (LFs) for end-of-line (EOL) and have no end-of-file (EOF) marker; Macintoshes use carriage returns (CRs) for EOLs; and most PC operating systems use CR+LF for EOLs and control-Z for EOF. In addition, IBM mainframes and the Michigan Terminal System use EBCDIC rather than the more common ASCII character set, and NT supports Unicode.) Note that *zip*'s identification of text files is by no means perfect; some ''text'' files may actually be binary and vice versa. *unzip* therefore prints ''`[text]`'' or ''`[binary]`'' as a visual check for each file it extracts when using the **−a** option. The **−aa** option forces all files to be extracted as text, regardless of the supposed file type. On VMS, see also **−S**.

−b [general] treat all files as binary (no text conversions). This is a shortcut for**−−−a**.

−b [Tandem] force the creation files with filecode type 180 ('C') when extracting Zip entries marked as "text". (On Tandem, **−a** is enabled by default, see above).

−b [VMS] auto-convert binary files (see **−a** above) to fixed-length, 512-byte record format. Doubling the option (**−bb**) forces all files to be extracted in this format. When extracting to standard output (**−c** or **−p** option in effect), the default conversion of text record delimiters is disabled for binary (**−b**) resp. all (**−bb**) files.

−B [when compiled with UNIXBACKUP defined] save a backup copy of each overwritten file. The backup file is gets the name of the target file with a tilde and optionally a unique sequence number (up to 5 digits) appended. The sequence number is applied whenever another file with the original name plus tilde already exists. When used together with the "overwrite all" option **−o**, numbered backup files are never created. In this case, all backup files are named as the original file with an appended tilde, existing backup files are deleted without notice. This feature works similarly to the default behavior of *emacs*(1) in many locations.

Example: the old copy of ''`foo`'' is renamed to ''`foo~`''.

Warning: Users should be aware that the **-B** option does not prevent loss of existing data under all circumstances. For example, when *unzip* is run in overwrite-all mode, an existing ''`foo~`'' file is deleted before *unzip* attempts to rename ''`foo`'' to ''`foo~`''. When this rename attempt fails (because of a file locks, insufficient privileges, or ...), the extraction of ''`foo~`'' gets cancelled, but the old backup file is already lost. A similar scenario takes place when the sequence number range for numbered backup files gets exhausted (99999, or 65535 for 16-bit systems). In this case, the backup file with the maximum sequence number is deleted and replaced by the new backup version without notice.

−C use case-insensitive matching for the selection of archive entries from the command-line list of extract selection patterns. *unzip*' s philosophy is ''you get what you ask for'' (this is also responsible for the **−L**/**−U** change; see the relevant options below). Because some file systems are fully case-sensitive (notably those under the Unix operating system) and because both ZIP archives and *unzip* itself are portable across platforms, *unzip*'s default behavior is to match both wildcard and literal filenames case-sensitively. That is, specifying ''`makefile`'' on the command line will *only* match ''makefile'' in the archive, not ''Makefile'' or ''MAKEFILE'' (and similarly for wildcard specifications). Since this does not correspond to the behavior of many other operating/file

systems (for example, OS/2 HPFS, which preserves mixed case but is not sensitive to it), the **−C** option may be used to force all filename matches to be case-insensitive. In the example above, all three files would then match ‘‘`makefile`’’ (or ‘‘`make*`’’, or similar). The **−C** option affects file specs in both the normal file list and the excluded-file list (xlist).

Please note that the **−C** option does neither affect the search for the zipfile(s) nor the matching of archive entries to existing files on the extraction path. On a case-sensitive file system, *unzip* will never try to overwrite a file ‘‘FOO’’ when extracting an entry ‘‘foo’’!

−D skip restoration of timestamps for extracted items. Normally, *unzip* tries to restore all meta-information for extracted items that are supplied in the Zip archive (and do not require privileges or impose a security risk). By specifying **−D**, *unzip* is told to suppress restoration of timestamps for directories explicitly created from Zip archive entries. This option only applies to ports that support setting timestamps for directories (currently ATheOS, BeOS, MacOS, OS/2, Unix, VMS, Win32, for other *unzip* ports, **−D** has no effect). The duplicated option**−DD** forces suppression of timestamp restoration for all extracted entries (files and directories). This option results in setting the timestamps for all extracted entries to the current time.

On VMS, the default setting for this option is **−D** for consistency with the behaviour of BACKUP: file timestamps are restored, timestamps of extracted directories are left at the current time. To enable restoration of directory timestamps, the negated option **−-D** should be specified. On VMS, the option **−D** disables timestamp restoration for all extracted Zip archive items. (Here, a single **−D** on the command line combines with the default **−D** to do what an explicit **−DD** does on other systems.)

−E [MacOS only] display contents of MacOS extra field during restore operation.

−F [Acorn only] suppress removal of NFS filetype extension from stored filenames.

−F [non-Acorn systems supporting long filenames with embedded commas, and only if compiled with ACORN_FTYPE_NFS defined] translate filetype information from ACORN RISC OS extra field blocks into a NFS filetype extension and append it to the names of the extracted files. (When the stored filename appears to already have an appended NFS filetype extension, it is replaced by the info from the extra field.)

−i [MacOS only] ignore filenames stored in MacOS extra fields. Instead, the most compatible filename stored in the generic part of the entry’s header is used.

−j junk paths. The archive’s directory structure is not recreated; all files are deposited in the extraction directory (by default, the current one).

−J [BeOS only] junk file attributes. The file’s BeOS file attributes are not restored, just the file’s data.

−J [MacOS only] ignore MacOS extra fields. All Macintosh specific info is skipped. Data-fork and resource-fork are restored as separate files.

−K [AtheOS, BeOS, Unix only] retain SUID/SGID/Tacky file attributes. Without this flag, these attribute bits are cleared for security reasons.

−L convert to lowercase any filename originating on an uppercase-only operating system or file system. (This was *unzip*’s default behavior in releases prior to 5.11; the new default behavior is identical to the old behavior with the **−U** option, which is now obsolete and will be removed in a future release.) Depending on the archiver, files archived under single-case file systems (VMS, old MS-DOS FAT, etc.) may be stored as all-uppercase names; this can be ugly or inconvenient when extracting to a case-preserving file system such as OS/2 HPFS or a case-sensitive one such as under Unix. By default *unzip* lists and extracts such filenames exactly as they’re stored (excepting truncation, conversion of unsupported characters, etc.); this option causes the names of all files from certain systems to be converted to lowercase. The**−LL** option forces con version of every filename to lowercase, regardless of the originating file system.

−M pipe all output through an internal pager similar to the Unix *more*(1) command. At the end of a screenful of output, *unzip* pauses with a ‘‘−−More−−’’ prompt; the next screenful may be viewed

by pressing the Enter (Return) key or the space bar. *unzip* can be terminated by pressing the ‘‘q’’ key and, on some systems, the Enter/Return key. Unlike Unix *more*(1), there is no forward-searching or editing capability. Also,*unzip* doesn’ t notice if long lines wrap at the edge of the screen, effectively resulting in the printing of two or more lines and the likelihood that some text will scroll off the top of the screen before being viewed. On some systems the number of available lines on the screen is not detected, in which case *unzip* assumes the height is 24 lines.

−n never overwrite existing files. If a file already exists, skip the extraction of that file without prompting. By default *unzip* queries before extracting any file that already exists; the user may choose to overwrite only the current file, overwrite all files, skip extraction of the current file, skip extraction of all existing files, or rename the current file.

−N [Amiga] extract file comments as Amiga filenotes. File comments are created with the −c option of *zip*(1), or with the −N option of the Amiga port of *zip*(1), which stores filenotes as comments.

−o overwrite existing files without prompting. This is a dangerous option, so use it with care. (It is often used with **−f**, however, and is the only way to overwrite directory EAs under OS/2.)

−P *password*

use *password* to decrypt encrypted zipfile entries (if any). **THIS IS INSECURE!** Many multi-user operating systems provide ways for any user to see the current command line of any other user; even on stand-alone systems there is always the threat of over-the-shoulder peeking. Storing the plaintext password as part of a command line in an automated script is even worse. Whenever possible, use the non-echoing, interactive prompt to enter passwords. (And where security is truly important, use strong encryption such as Pretty Good Privacy instead of the relatively weak encryption provided by standard zipfile utilities.)

−q perform operations quietly (**−qq** = even quieter). Ordinarily *unzip* prints the names of the files it’s extracting or testing, the extraction methods, any file or zipfile comments that may be stored in the archive, and possibly a summary when finished with each archive. The**−q**[**q**] options suppress the printing of some or all of these messages.

−s [OS/2, NT, MS-DOS] convert spaces in filenames to underscores. Since all PC operating systems allow spaces in filenames, *unzip* by default extracts filenames with spaces intact (e.g., ‘‘`EA DATA. SF`’’). This can be awkward, however, since MS-DOS in particular does not gracefully support spaces in filenames. Conversion of spaces to underscores can eliminate the awkwardness in some cases.

−S [VMS] convert text files (**−a**, **−aa**) into Stream_LF record format, instead of the text-file default, variable-length record format. (Stream_LF is the default record format of VMS *unzip*. It is applied unless conversion (**−a**, **−aa** and/or **−b**, **−bb**) is requested or a VMS-specific entry is processed.)

−U [UNICODE_SUPPORT only] modify or disable UTF-8 handling. When UNICODE_SUPPORT is available, the option **−U** forces *unzip* to escape all non-ASCII characters from UTF-8 coded filenames as ‘‘#Uxxxx’’ (for UCS-2 characters, or ‘‘#Lxxxxxx’’ for unicode codepoints needing 3 octets). This option is mainly provided for debugging purpose when the fairly new UTF-8 support is suspected to mangle up extracted filenames.

The option **−UU** allows to entirely disable the recognition of UTF-8 encoded filenames. The handling of filename codings within *unzip* falls back to the behaviour of previous versions.

[old, obsolete usage] leave filenames uppercase if created under MS-DOS, VMS, etc. See **−L** above.

−V retain (VMS) file version numbers. VMS files can be stored with a version number, in the format `file.ext;##`. By default the ‘‘`;##`’’ version numbers are stripped, but this option allows them to be retained. (On file systems that limit filenames to particularly short lengths, the version numbers may be truncated or stripped regardless of this option.)

−W [only when WILD_STOP_AT_DIR compile-time option enabled] modifies the pattern matching routine so that both ‘?’ (single-char wildcard) and ‘*’ (multi-char wildcard) do not match the directory separator character ‘/’. (The two-character sequence ‘‘**’’ acts as a multi-char wildcard

that includes the directory separator in its matched characters.) Examples:

```
"*.c" matches "foo.c" but not "mydir/foo.c"
"**.c" matches both "foo.c" and "mydir/foo.c"
"*/*.c" matches "bar/foo.c" but not "baz/bar/foo.c"
"??*/*" matches "ab/foo" and "abc/foo"
        but not "a/foo" or "a/b/foo"
```

This modified behaviour is equivalent to the pattern matching style used by the shells of some of UnZip's supported target OSs (one example is Acorn RISC OS). This option may not be available on systems where the Zip archive's internal directory separator character '/' is allowed as regular character in native operating system filenames. (Currently, UnZip uses the same pattern matching rules for both wildcard zipfile specifications and zip entry selection patterns in most ports. For systems allowing '/' as regular filename character, the -W option would not work as expected on a wildcard zipfile specification.)

−X [VMS, Unix, OS/2, NT, Tandem] restore owner/protection info (UICs and ACL entries) under VMS, or user and group info (UID/GID) under Unix, or access control lists (ACLs) under certain network-enabled versions of OS/2 (Warp Server with IBM LAN Server/Requester 3.0 to 5.0; Warp Connect with IBM Peer 1.0), or security ACLs under Windows NT. In most cases this will require special system privileges, and doubling the option (**−XX**) under NT instructs *unzip* to use privileges for extraction; but under Unix, for example, a user who belongs to several groups can restore files owned by any of those groups, as long as the user IDs match his or her own. Note that ordinary file attributes are always restored--this option applies only to optional, extra ownership info available on some operating systems. [NT's access control lists do not appear to be especially compatible with OS/2's, so no attempt is made at cross-platform portability of access privileges. It is not clear under what conditions this would ever be useful anyway.]

−Y [VMS] treat archived file name endings of ''.nnn'' (where ''nnn'' is a decimal number) as if they were VMS version numbers (';nnn''). (The default is to treat them as file types.) Example:

```
"a.b.3" -> "a.b;3".
```

−$** [MS-DOS, OS/2, NT] restore the volume label if the extraction medium is removable (e.g., a diskette). Doubling the option (**−$$) allows fixed media (hard disks) to be labelled as well. By default, volume labels are ignored.

−/ *extensions*

[Acorn only] overrides the extension list supplied by Unzip$Ext environment variable. During extraction, filename extensions that match one of the items in this extension list are swapped in front of the base name of the extracted file.

−: [all but Acorn, VM/CMS, MVS, Tandem] allows to extract archive members into locations outside of the current '' extraction root folder''. For security reasons, *unzip* normally removes ''parent dir'' path components (''../'') from the names of extracted file. This safety feature (new for version 5.50) prevents *unzip* from accidentally writing files to ''sensitive'' areas outside the active extraction folder tree head. The **−:** option lets *unzip* switch back to its previous, more liberal behaviour, to allow exact extraction of (older) archives that used ''../'' components to create multiple directory trees at the level of the current extraction folder. This option does not enable writing explicitly to the root directory (''/''). To achieve this, it is necessary to set the extraction target folder to root (e.g. **−d /**). However, when the **−:** option is specified, it is still possible to implicitly write to the root directory by specifying enough ''../'' path components within the zip archive. Use this option with extreme caution.

−^ [Unix only] allow control characters in names of extracted ZIP archive entries. On Unix, a file name may contain any (8-bit) character code with the two exception '/' (directory delimiter) and NUL (0x00, the C string termination indicator), unless the specific file system has more restrictive conventions. Generally, this allows to embed ASCII control characters (or even sophisticated control sequences) in file names, at least on 'native' Unix file systems. However, it may be highly suspicious to make use of this Unix "feature". Embedded control characters in file names might

have nasty side effects when displayed on screen by some listing code without sufficient filtering. And, for ordinary users, it may be difficult to handle such file names (e.g. when trying to specify it for open, copy, move, or delete operations). Therefore, *unzip* applies a filter by default that removes potentially dangerous control characters from the extracted file names. The **-^** option allows to override this filter in the rare case that embedded filename control characters are to be intentionally restored.

−2 [VMS] force unconditionally conversion of file names to ODS2-compatible names. The default is to exploit the destination file system, preserving case and extended file name characters on an ODS5 destination file system; and applying the ODS2-compatibility file name filtering on an ODS2 destination file system.

ENVIRONMENT OPTIONS

unzip's default behavior may be modified via options placed in an environment variable. This can be done with any option, but it is probably most useful with the **−a**, **−L**, **−C**, **−q**, **−o**, or **−n** modifiers: make *unzip* auto-convert text files by default, make it convert filenames from uppercase systems to lowercase, make it match names case-insensitively, make it quieter, or make it always overwrite or never overwrite files as it extracts them. For example, to make *unzip* act as quietly as possible, only reporting errors, one would use one of the following commands:

Unix Bourne shell:
UNZIP=−qq; export UNZIP

Unix C shell:
setenv UNZIP −qq

OS/2 or MS-DOS:
set UNZIP=−qq

VMS (quotes for *lowercase*):
define UNZIP_OPTS "−qq"

Environment options are, in effect, considered to be just like any other command-line options, except that they are effectively the first options on the command line. To override an environment option, one may use the ‘‘minus operator’’ to remove it. For instance, to override one of the quiet-flags in the example above, use the command

```
unzip −−q[other options] zipfile
```

The first hyphen is the normal switch character, and the second is a minus sign, acting on the q option. Thus the effect here is to cancel one quantum of quietness. To cancel both quiet flags, two (or more) minuses may be used:

```
unzip −t−−q zipfile
unzip −−−qt zipfile
```

(the two are equivalent). This may seem awkward or confusing, but it is reasonably intuitive: just ignore the first hyphen and go from there. It is also consistent with the behavior of Unix *nice*(1).

As suggested by the examples above, the default variable names are UNZIP_OPTS for VMS (where the symbol used to install *unzip* as a foreign command would otherwise be confused with the environment variable), and UNZIP for all other operating systems. For compatibility with *zip*(1), UNZIPOPT is also accepted (don't ask). If both UNZIP and UNZIPOPT are defined, however, UNZIP takes precedence. *unzip*' s diagnostic option (**−v** with no zipfile name) can be used to check the values of all four possible *unzip* and *zipinfo* environment variables.

The timezone variable (TZ) should be set according to the local timezone in order for the **−f** and **−u** to operate correctly. See the description of**−f** abo ve for details. This variable may also be necessary to get timestamps of extracted files to be set correctly. The WIN32 (Win9x/ME/NT4/2K/XP/2K3) port of *unzip* gets the timezone configuration from the registry, assuming it is correctly set in the Control Panel. The TZ variable is ignored for this port.

DECRYPTION

Encrypted archives are fully supported by Info-ZIP software, but due to United States export restrictions, de-/encryption support might be disabled in your compiled binary. However, since spring 2000, US export restrictions have been liberated, and our source archives do now include full crypt code. In case you need binary distributions with crypt support enabled, see the file ''WHERE'' in any Info-ZIP source or binary distribution for locations both inside and outside the US.

Some compiled versions of *unzip* may not support decryption. To check a version for crypt support, either attempt to test or extract an encrypted archive, or else check *unzip*'s diagnostic screen (see the **−v** option above) for ''`[decryption]`'' as one of the special compilation options.

As noted above, the **−P** option may be used to supply a password on the command line, but at a cost in security. The preferred decryption method is simply to extract normally; if a zipfile member is encrypted, *unzip* will prompt for the password without echoing what is typed. *unzip* continues to use the same passw ord as long as it appears to be valid, by testing a 12-byte header on each file. The correct password will always check out against the header, but there is a 1-in-256 chance that an incorrect password will as well. (This is a security feature of the PKWARE zipfile format; it helps prevent brute-force attacks that might otherwise gain a large speed advantage by testing only the header.) In the case that an incorrect password is given but it passes the header test anyway, either an incorrect CRC will be generated for the extracted data or else *unzip* will fail during the extraction because the ''decrypted'' bytes do not constitute a valid compressed data stream.

If the first password fails the header check on some file, *unzip* will prompt for another password, and so on until all files are extracted. If a password is not known, entering a null password (that is, just a carriage return or ''Enter'') is taken as a signal to skip all further prompting. Only unencrypted files in the archive(s) will thereafter be extracted. (In fact, that's not quite true; older versions of *zip*(1) and *zipcloak*(1) allowed null passwords, so *unzip* checks each encrypted file to see if the null password works. This may result in ''false positives'' and extraction errors, as noted above.)

Archives encrypted with 8-bit passwords (for example, passwords with accented European characters) may not be portable across systems and/or other archivers. This problem stems from the use of multiple encoding methods for such characters, including Latin-1 (ISO 8859-1) and OEM code page 850. DOS *PKZIP* 2.04g uses the OEM code page; Windows *PKZIP* 2.50 uses Latin-1 (and is therefore incompatible with DOS *PKZIP*); Info-ZIP uses the OEM code page on DOS, OS/2 and Win3.x ports but ISO coding (Latin-1 etc.) everywhere else; and Nico Mak's *WinZip* 6.x does not allow 8-bit passwords at all. *UnZip* 5.3 (or newer) attempts to use the default character set first (e.g., Latin-1), followed by the alternate one (e.g., OEM code page) to test passwords. On EBCDIC systems, if both of these fail, EBCDIC encoding will be tested as a last resort. (EBCDIC is not tested on non-EBCDIC systems, because there are no known archivers that encrypt using EBCDIC encoding.) ISO character encodings other than Latin-1 are not supported. The new addition of (partially) Unicode (resp. UTF-8) support in *UnZip* 6.0 has not yet been adapted to the encryption password handling in *unzip*. On systems that use UTF-8 as native character encoding, *unzip* simply tries decryption with the native UTF-8 encoded password; the built-in attempts to check the password in translated encoding have not yet been adapted for UTF-8 support and will consequently fail.

EXAMPLES

To use *unzip* to extract all members of the archive *letters.zip* into the current directory and subdirectories below it, creating any subdirectories as necessary:

```
unzip letters
```

To extract all members of *letters.zip* into the current directory only:

```
unzip -j letters
```

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

```
unzip -tq letters
```

To test *all* zipfiles in the current directory, printing only the summaries:

```
unzip -tq \*.zip
```

(The backslash before the asterisk is only required if the shell expands wildcards, as in Unix; double quotes could have been used instead, as in the source examples below.) To extract to standard output all members of *letters.zip* whose names end in *.tex*, auto-converting to the local end-of-line convention and piping the output into *more*(1):

```
unzip −ca letters \*.tex | more
```

To extract the binary file *paper1.dvi* to standard output and pipe it to a printing program:

```
unzip −p articles paper1.dvi | dvips
```

To extract all FORTRAN and C source files--*.f, *.c, *.h, and Makefile--into the /tmp directory:

```
unzip source.zip "*.[fch]" Makefile -d /tmp
```

(the double quotes are necessary only in Unix and only if globbing is turned on). To extract all FORTRAN and C source files, regardless of case (e.g., both *.c and *.C, and any makefile, Makefile, MAKEFILE or similar):

```
unzip −C source.zip "*.[fch]" makefile -d /tmp
```

To extract any such files but convert any uppercase MS-DOS or VMS names to lowercase and convert the line-endings of all of the files to the local standard (without respect to any files that might be marked ‘‘binary’’):

```
unzip −aaCL source.zip "*.[fch]" makefile -d /tmp
```

To extract only newer versions of the files already in the current directory, without querying (NOTE: be careful of unzipping in one timezone a zipfile created in another--ZIP archives other than those created by Zip 2.1 or later contain no timezone information, and a ‘‘newer’’ file from an eastern timezone may, in fact, be older):

```
unzip −fo sources
```

To extract newer versions of the files already in the current directory and to create any files not already there (same caveat as previous example):

```
unzip −uo sources
```

To display a diagnostic screen showing which *unzip* and *zipinfo* options are stored in environment variables, whether decryption support was compiled in, the compiler with which *unzip* was compiled, etc.:

```
unzip −v
```

In the last five examples, assume that UNZIP or UNZIP_OPTS is set to -q. To do a singly quiet listing:

```
unzip −l file.zip
```

To do a doubly quiet listing:

```
unzip −ql file.zip
```

(Note that the ‘‘`.zip`’’ is generally not necessary.) To do a standard listing:

```
unzip −−ql file.zip
```
or
```
unzip −l−q file.zip
```
or
```
unzip −l−−q file.zip
```
(Extra minuses in options don’t hurt.)

TIPS

The current maintainer, being a lazy sort, finds it very useful to define a pair of aliases: `tt` for ‘‘`unzip −tq`’’ and `ii` for ‘‘`unzip −Z`’’ (or ‘‘`zipinfo`’’). One may then simply type ‘‘`tt zipfile`’’ to test an archive, something that is worth making a habit of doing. With luck *unzip* will report ‘‘`No errors detected in compressed data of zipfile.zip`,’’ after which one may breathe a sigh of relief.

The maintainer also finds it useful to set the UNZIP environment variable to ‘‘`−aL`’’ and is tempted to add

‘‘`-C`’’ as well. His ZIPINFO variable is set to ‘‘`-z`’’.

DIAGNOSTICS

The exit status (or error level) approximates the exit codes defined by PKWARE and takes on the following values, except under VMS:

0	normal; no errors or warnings detected.
1	one or more warning errors were encountered, but processing completed successfully anyway. This includes zipfiles where one or more files was skipped due to unsupported compression method or encryption with an unknown password.
2	a generic error in the zipfile format was detected. Processing may have completed successfully anyway; some broken zipfiles created by other archivers have simple workarounds.
3	a severe error in the zipfile format was detected. Processing probably failed immediately.
4	*unzip* was unable to allocate memory for one or more buffers during program initialization.
5	*unzip* was unable to allocate memory or unable to obtain a tty to read the decryption password(s).
6	*unzip* was unable to allocate memory during decompression to disk.
7	*unzip* was unable to allocate memory during in-memory decompression.
8	[currently not used]
9	the specified zipfiles were not found.
10	invalid options were specified on the command line.
11	no matching files were found.
50	the disk is (or was) full during extraction.
51	the end of the ZIP archive was encountered prematurely.
80	the user aborted *unzip* prematurely with control-C (or similar)
81	testing or extraction of one or more files failed due to unsupported compression methods or unsupported decryption.
82	no files were found due to bad decryption password(s). (If even one file is successfully processed, however, the exit status is 1.)

VMS interprets standard Unix (or PC) return values as other, scarier-looking things, so *unzip* instead maps them into VMS-style status codes. The current mapping is as follows: 1 (success) for normal exit, 0x7fff0001 for warning errors, and (0x7fff000? + 16*normal_unzip_exit_status) for all other errors, where the ‘?’ is 2 (error) for *unzip* values 2, 9-11 and 80-82, and 4 (fatal error) for the remaining ones (3-8, 50, 51). In addition, there is a compilation option to expand upon this behavior: defining RETURN_CODES results in a human-readable explanation of what the error status means.

BUGS

Multi-part archives are not yet supported, except in conjunction with *zip*. (All parts must be concatenated together in order, and then ‘‘`zip -F`’’ (for *zip 2.x*) or ‘‘`zip -FF`’’ (for *zip 3.x*) must be performed on the concatenated archive in order to ‘‘fix’’ it. Also, *zip 3.0* and later can combine multi-part (split) archives into a combined single-file archive using ‘‘`zip -s- inarchive -O outarchive`’’. See the*zip 3* manual page for more information.) This will definitely be corrected in the next major release.

Archives read from standard input are not yet supported, except with *funzip* (and then only the first member of the archive can be extracted).

Archives encrypted with 8-bit passwords (e.g., passwords with accented European characters) may not be portable across systems and/or other archivers. See the discussion in**DECR YPTION** above.

unzip's **−M** (''more'') option tries to take into account automatic wrapping of long lines. However, the code may fail to detect the correct wrapping locations. First, TAB characters (and similar control sequences) are not taken into account, they are handled as ordinary printable characters. Second, depending on the actual system / OS port, *unzip* may not detect the true screen geometry but rather rely on "commonly used" default dimensions. The correct handling of tabs would require the implementation of a query for the actual tabulator setup on the output console.

Dates, times and permissions of stored directories are not restored except under Unix. (On Windows NT and successors, timestamps are now restored.)

[MS-DOS] When extracting or testing files from an archive on a defective floppy diskette, if the ''Fail'' option is chosen from DOS's ''Abort, Retry, Fail?'' message, older versions of *unzip* may hang the system, requiring a reboot. This problem appears to be fixed, but control-C (or control-Break) can still be used to terminate *unzip*.

Under DEC Ultrix, *unzip* would sometimes fail on long zipfiles (bad CRC, not always reproducible). This was apparently due either to a hardware bug (cache memory) or an operating system bug (improper handling of page faults?). Since Ultrix has been abandoned in favor of Digital Unix (OSF/1), this may not be an issue anymore.

[Unix] Unix special files such as FIFO buffers (named pipes), block devices and character devices are not restored even if they are somehow represented in the zipfile, nor are hard-linked files relinked. Basically the only file types restored by *unzip* are regular files, directories and symbolic (soft) links.

[OS/2] Extended attributes for existing directories are only updated if the **−o** (''overwrite all'') option is given. This is a limitation of the operating system; because directories only have a creation time associated with them, *unzip* has no way to determine whether the stored attributes are newer or older than those on disk. In practice this may mean a two-pass approach is required: first unpack the archive normally (with or without freshening/updating existing files), then overwrite just the directory entries (e.g., ''`unzip -o foo */`'').

[VMS] When extracting to another directory, only the *[.foo]* syntax is accepted for the **−d** option; the simple Unix *foo* syntax is silently ignored (as is the less common VMS *foo.dir* syntax).

[VMS] When the file being extracted already exists, *unzip*'s query only allows skipping, overwriting or renaming; there should additionally be a choice for creating a new version of the file. In fact, the ''overwrite'' choice does create a new version; the old version is not overwritten or deleted.

SEE ALSO

funzip(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

The primary Info-ZIP authors (current semi-active members of the Zip-Bugs workgroup) are: Ed Gordon (Zip, general maintenance, shared code, Zip64, Win32, Unix, Unicode); Christian Spieler (UnZip maintenance coordination, VMS, MS-DOS, Win32, shared code, general Zip and UnZip integration and optimization); Onno van der Linden (Zip); Mike White (Win32, Windows GUI, Windows DLLs); Kai Uwe Rommel (OS/2, Win32); Steven M. Schweda (VMS, Unix, support of new features); Paul Kienitz (Amiga, Win32, Unicode); Chris Herborth (BeOS, QNX, Atari); Jonathan Hudson (SMS/QDOS); Sergio Monesi (Acorn RISC OS); Harald Denker (Atari, MVS); John Bush (Solaris, Amiga); Hunter Goatley (VMS, Info-ZIP Site maintenance); Steve Salisbury (Win32); Steve Miller (Windows CE GUI), Johnny Lee (MS-DOS, Win32, Zip64); and Dave Smith (Tandem NSK).

The following people were former members of the Info-ZIP development group and provided major contributions to key parts of the current code: Greg ''Cave Newt'' Roelofs (UnZip, unshrink decompression); Jean-loup Gailly (deflate compression); Mark Adler (inflate decompression, fUnZip).

The author of the original unzip code upon which Info-ZIP's was based is Samuel H. Smith; Carl Mascott did the first Unix port; and David P. Kirschbaum organized and led Info-ZIP in its early days with Keith Petersen hosting the original mailing list at WSMR-SimTel20. The full list of contributors to UnZip has grown quite large; please refer to the CONTRIBS file in the UnZip source distribution for a relatively complete version.

VERSIONS

v1.2	15 Mar 89	Samuel H. Smith
v2.0	9 Sep 89	Samuel H. Smith
v2.x	fall 1989	many Usenet contributors
v3.0	1 May 90	Info-ZIP (DPK, consolidator)
v3.1	15 Aug 90	Info-ZIP (DPK, consolidator)
v4.0	1 Dec 90	Info-ZIP (GRR, maintainer)
v4.1	12 May 91	Info-ZIP
v4.2	20 Mar 92	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.0	21 Aug 92	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.01	15 Jan 93	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.1	7 Feb 94	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.11	2 Aug 94	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.12	28 Aug 94	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.2	30 Apr 96	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.3	22 Apr 97	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.31	31 May 97	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.32	3 Nov 97	Info-ZIP (Zip-Bugs subgroup, GRR)
v5.4	28 Nov 98	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.41	16 Apr 00	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.42	14 Jan 01	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.5	17 Feb 02	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.51	22 May 04	Info-ZIP (Zip-Bugs subgroup, SPC)
v5.52	28 Feb 05	Info-ZIP (Zip-Bugs subgroup, SPC)
v6.0	20 Apr 09	Info-ZIP (Zip-Bugs subgroup, SPC)