zlib

zlib is a software library used for data compression. zlib was written by Jean-Loup Gailly and Mark Adler and is an abstraction of the DEFLATE compression algorithm used in their gzip file compression program. zlib is also a crucial component of many software platforms including Linux, Mac OS X, and iOS. It has also been used in gaming consoles such as the PlayStation 4, PlayStation 3, Wii U, Wii, Xbox One and Xbox 360.

The first public version of zlib, 0.9, was released on 1 May 1995 and was originally intended for use with the libpng image library. It is free software, distributed under the zlib license.

1 Capabilities

1.1 Encapsulation

zlib compressed data are typically written with a gzip or a zlib wrapper. The wrapper encapsulates the raw DEFLATE data by adding a header and trailer. This provides stream identification and error detection that are not provided by the raw DEFLATE data.

The gzip header is larger than the zlib header, as it stores a file name and other file system information. This is the header format used in the ubiquitous gzip file format.

1.2 Algorithm

As of February 2010, zlib only supports one algorithm called DEFLATE, that is a variation of LZ77 (Lempel–Ziv 1977). This algorithm provides good compression on a wide variety of data with minimal use of system resources. This is also the algorithm used in the ZIP archive format.

The header makes allowance for other algorithms, but none are currently implemented.

1.3 Resource use

zlib provides facilities for control of processor and memory use. A compression level value may be supplied that trades-off speed with compression. There are also facilities for conserving memory, useful in restricted memory environments such as some embedded systems.

1.4 Strategy

The compression can be optimized for specific types of data. If one is using the library to always compress specific types of data, then using a specific strategy may improve compression and performance. For example, if the data contain long lengths of repeated bytes, the RLE (run-length encoding) strategy may give good results at higher speed. For general data, the default strategy is preferred.

1.5 Error handling

Errors in compressed data may be detected and skipped. Further, if "full-flush" points are written to the compressed stream, then corrupt data can be skipped, and the decompression will resynchronize at the next flush point - although no error recovery of the corrupt data is provided. Full-flush points are useful for large data streams on unreliable channels, where some data loss is unimportant, such as in some multimedia applications. However, creating many flush points can affect the speed as well as amount (ratio) of compression.

1.6 Data length

There is no limit to the length of data that can be compressed or decompressed. Repeated calls to the library allow an unlimited numbers of blocks of data to be handled. Some ancillary code (counters) may suffer from overflow for long data streams, but this does not affect the actual compression or decompression.

When compressing a long (or infinite) data stream, it is advisable to write regular full-flush points.

2 Applications

Today, zlib is something of a *de facto* standard, to the point that zlib and DEFLATE are often used interchangeably in standards documents, with thousands of applications relying on it for compression, either directly or indirectly.[1] These include:

- The Linux kernel, where zlib is used to implement compressed network protocols, compressed file systems, and to decompress the kernel image at boot time.

- libpng, the reference implementation for the PNG image format, which specifies DEFLATE as the stream compression for its bitmap data.
- libwww, an API for Web applications like Web browsers.
- The Apache HTTP server, which uses zlib to implement HTTP/1.1.
- The OpenSSH client and server, which rely on zlib to perform the optional compression offered by the Secure Shell protocol.
- The OpenSSL and GnuTLS security libraries, which can optionally use zlib to compress TLS connections.
- The FFmpeg multimedia library, which uses zlib to read and write the DEFLATE-compressed parts of stream formats such as Matroska.
- The rsync remote file synchronizer, which uses zlib to implement optional protocol compression.
- The dpkg and RPM package managers, which use zlib to unpack files from compressed software packages.
- The Subversion and CVS version control systems, which use zlib to compress traffic to and from remote repositories.
- The Git version control system uses zlib to store the contents of its data objects (blobs, trees, commits and tags).
- The PostgreSQL RDBMS that uses zlib with custom dump format (pg_dump -Fc) for database backups.
- The class System.IO.Compression.DeflateStream of the Microsoft .NET Framework 2.0 and higher.[2]
- The "deflate" utility in TORNADO as part of VxWorks Operating System made by Wind River Systems uses zlib to compress boot ROM images.

zlib is also used in many embedded devices, such as the Apple iPhone and Sony PlayStation 3, because the code is portable, liberally licensed, and has a relatively small memory footprint.

3 See also

- DEFLATE
- gzip
- LZ77
- ZIP (file format)
- zlib License

4 References

[1] Gailly, Jean-loup; Adler, Mark (2002-04-18), *zlib Applications*

[2] System.IO.Compression.DeflateStream. MSDN Library.

5 External links

- Official website
- RFC 1950—ZLIB Compressed Data Format
- RFC 1951—DEFLATE Compressed Data Format
- RFC 1952—GZIP file format
- miniz.c, a single-source-file, public-domain implementation of the zlib API for embedded and memory-limited use cases

6 Text and image sources, contributors, and licenses

6.1 Text

- **Zlib** *Source:* https://en.wikipedia.org/wiki/Zlib?oldid=669002784 *Contributors:* Ghakko, BL~enwiki, Nknight, CesarB, Greenrd, Furykef, JesseW, Fleminra, Oneiros, Squash, Thorwald, Rich Farmbrough, Pmsyyz, Tarjei Knapstad, Smyth, Abelson, Horkana, Gronky, Danakil, John Vandenberg, Minghong, Towel401, CyberSkull, Apoc2400, Danhash, Tony Sidaway, K3rb, Cristan, Karnesky, Qwertyus, Dpv, Abizern, Hellothatsme~enwiki, Scandum, Strait, KamasamaK, Reinis, Yuletide, FlaBot, Chobot, YurikBot, Hydrargyrum, Avraham, Pegship, ReCover, CWenger, JLaTondre, DmitriyV, SmackBot, Thumperward, Morte, Kimero, Jerome Charles Potts, Frap, VMS Mosaic, A5b, Guyjohnston, Kvng, Arto B, CRGreathouse, Preacher Bob, Requestion, Bobblehead, Magioladitis, Japo, Kunaldeo, Speck-Made, Genium, Kahlil88, Azimout, Free Software Knight, Kl4m-AWB, Iuhkjhk87y678, Socrates2008, Dradler, Mabdul, Scientus, Rkhardy, OlEnglish, Fryed-peach, Legobot, Yobot, Wickorama, ArthurBot, NorwalkJames, APhillips801, Sae1962, Kem wiki, Liamzebedee, Gszpetkowski, ZéroBot, Anir1uph, Bomazi, Senator2029, DanielZazula, Rjs.swarnkar, Flylo, Jimw338, Dobie80, EfiHerbst and Anonymous: 38

6.2 Images

- **File:Free_Software_Portal_Logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/6/67/Nuvola_apps_emacs_vector.svg *License:* LGPL *Contributors:*
- Nuvola_apps_emacs.png *Original artist:* Nuvola_apps_emacs.png: David Vignoni
- **File:Zlib_3D_green.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/5f/Zlib_3D_green.png *License:* ZLIB *Contributors:* ? *Original artist:* ?

6.3 Content license

- Creative Commons Attribution-Share Alike 3.0