SWI-Prolog binding to zlib

Jan Wielemaker
University of Amsterdam
VU University Amsterdam
The Netherlands
E-mail: J.Wielemaker@vu.nl

September 25, 2017

Abstract

The library zlib provides a binding to the zlib general purpose compression library. The prolog library aims as seamlessly reading and writing files compatible to the gzip program as well as compressed (network) communication.

Contents

1 Zlib and compression

Zlib is a widespread library implementing the RFC1950 (zlib wrapper), RFC1951 (deflate stream) and RFC1952 (gzip wrapper) compression standards. The SWI-Prolog binding is a foreign library that creates a compressed stream as a wrapper around a normal stream. Implemented this way, it can perform a wide variety of tasks:

- Read/write gzip compatible files
- Setup standard compressed stream communication
- Realise in-memory compression or decompression
- Deal with streams holding embedded compressed objects

The core predicate of the library is `zopen/3`. The remainder of the functionality of `zlib` is defined in Prolog and can be used as a starting point for other high-level primitives. See also `ztest.pl` providing test and demo code. This file is part of the source distribution.

Part of the functionality of this library can also be realised using the pipe interface and the `gzip` program. For example, a gziped file can also be opened in Prolog using the code below.

```
...
open(pipe('gunzip < file.gz'), read, In),
...
```

The advantage of this library over using an external program for such tasks is enhanced platform independence and reduced time to open a file. Platform independence is improved as we do not have to worry about availability of the `gunzip` utility and we do not have to worry about shell and filename quoting issues. While the above replacement code works well on most modern Unix systems, it only works with special precautions on Windows.[1]

The library becomes unavoidable if we consider compressed network communication. Here we get the stream from `tcp_open_socket/3`. The library provides efficient creation of a compressed stream, as well as support for flushing output through the standard Prolog `flush_output/1` call.

2 Predicate reference

zopen(+*Stream, -ZStream, +Options*)

Creates *ZStream*, providing compressed access to *Stream*. If an input stream is wrapped, it recognises a gzip or deflate header. If an output stream is wrapped, *Options* define the desired wrapper and compression level. The new *ZStream* inherits its *encoding* from *Stream*. In other words, if *Stream* is a text-stream, so is *ZStream*. The original *Stream* is switched to binary mode while it is wrapped. The original encoding of *Stream* is restored if *ZStream* is closed. Note that `zopen/3` does not actually process any data and therefore succeeds on input streams that do not contain valid data. Errors may be generated by read operations performed on the stream.

Defined options on output streams are:

[1] Install gunzip, deal with Windows path-names, the windows shell and quoting.

format(+*Format*)

Either `deflate` (default), `raw_deflate` or `gzip`. The `deflate` envelope is simple and short and is typically used for compressed (network) communication. The `raw_deflate` does not include an envelope and is often used as a step in crypographic encodings. The `gzip` envelope is compatible to the `gzip` program and intended to read/write compressed files.

level(+*Level*)

Number between 0 and 9, specifying the compression level, Higher levels use more resources. Default is 6, generally believed to be a good compromise between speed, memory requirement and compression.

multi_part(+*Boolean*)

If `true`, restart reading if the input is not at end-of-file. The default is `true` for gzip streams.

Generic options are:

close_parent(*Bool*)

If `true` (default), closing the compressed stream also closes (and thus invalidates) the wrapped stream. If `false`, the wrapped stream is *not* closed. This can be used to read/write a compressed data block as partial input/output on a stream.

gzopen(+*File, +Mode, -Stream*)

Same as `gzopen`(*File, Mode, Stream, []*).

gzopen(+*File, +Mode, -Stream, +Options*)

Open `gzip` compatible *File* for reading or writing. If a file is opened in =append= mode, a new gzip image will be added to the end of the file. The gzip standard defines that a file can hold multiple gzip images and inflating the file results in a concatenated stream of all inflated images. Options are passed to `open/4` and `zopen/3`. Default format is `gzip`.

3 Interaction with Prolog stream predicates

Using `flush_output/1` on a compressed stream causes a `Z_SYNC_FLUSH` on the stream. Using `close/1` on a compressed stream causes a `Z_FINISH` on the stream. If the stream uses the `gzip` format, a `gzip` compatible footer is written to the stream. If `close_parent` is set (default) the underlying stream is closed too. Otherwise it remains open and the user can continue communication in non-compressed format or reopen the stream for compression using `zopen/3`.

Index