

# Compress::Zlib

Alzi la mano chi non ha mai utilizzato [<http://www.winzip.com>|winzip], [<http://www.gzip.org>|gzip], [<http://www.rarlab.com>|rar] o il piu' recente [<http://sources.redhat.com/bzip2>|bzip2]!



La compressione dei dati è indubbiamente un argomento molto stimolante, e [<http://search.cpan.org/>|CPAN] ha una [[http://www.cpan.org/modules/by-category/17\\_Archiving\\_and\\_Compression/](http://www.cpan.org/modules/by-category/17_Archiving_and_Compression/)|categoria] a parte soltanto per questo, di cui probabilmente [CompressZlib|Compress::Zlib] è il modulo più universalmente noto e utilizzato. Come dice il nome si basa sulla libreria [<http://www.zlib.org>|zlib], quanto di più multiplatforma, portabile e collaudato si conosca da molti anni a questa parte.

[CompressZlib|Compress::Zlib], ormai alla versione 1.33, da poco tempo ingloba nella propria distribuzione anche i sorgenti di [<http://www.zlib.org>|zlib], a tutto vantaggio della facilità di installazione (ovviamente è richiesto un ambiente di sviluppo funzionante).

L'interfaccia del modulo è abbastanza 'polivalente', nel senso che ci sono diversi set di subroutine alternativi tra di loro che realizzano le stesse funzionalità. Questo all'inizio può confondere, ma per cominciare a lavorare sul serio ci basta poco.

Vediamo ad esempio come comprimere e decomprimere un buffer direttamente in memoria:

```
use Compress::Zlib;
my $original      = '<Your string here>';
my $compressed    = compress($original);
my $uncompressed = uncompress($compressed);

print "Compress::Zlib ", (
    $original eq $uncompressed
    ? "funziona!"
    : "NON FUNZIONA?!?!?"
), "\n";
```

Per chi conosce bene la zlib, [CompressZlib|Compress::Zlib] espone una serie di wrapper che ne ricalcano molto fedelmente il funzionamento (vedi `inflate*`(), `deflate*`()). Tuttavia Perl non sarebbe Perl se non rendesse le cose 'molto' più semplici :-)

Ecco un gzip dei poveri:

```
#!/usr/bin/perl
#
# Uso: gzip.pl <nome_file>
#
# Fa quasi esattamente quello che fa anche gzip
# e ovviamente legge gli stessi file!

use strict;
use warnings;
use Compress::Zlib qw(Z_FINISH);
use constant BUFFER_SIZE => 64 * 1024;

# Nome file come argomento
my $file = shift @ARGV;

# Se il file termina con .gz, decomprimi
```

```

if( rindex($file, '.gz') > 0 ) {
    unzip( $file, substr($file, 0, -3) );
} else {
    zip( $file, $file.'.gz' );
}

```

Beh, non è proprio così facile, ma quasi. Mancano le due funzioni zip() e unzip(), che fanno parte del programma di esempio:

```

# Decomprime il file sorgente scrivendo nel file destinazione
sub unzip {
    my($srcf, $dstf) = @_;

    if( open OUT, '>'.$dstf ) {
        binmode OUT;
        # Prealloca il buffer di lettura
        my $buffer = 0x00 x BUFFER_SIZE;
        if( my $gz = Compress::Zlib::gzopen( $srcf, 'rb' ) ) {
            print OUT $buffer while $gz->gzread( $buffer, BUFFER_SIZE );
            close OUT;
            $gz->gzclose();
            return 1;
        }
    }
    return 0;
}

# Comprime il file sorgente scrivendo nel file destinazione
sub zip {
    my($srcf, $dstf) = @_;

    if( open(IN, '<'.$srcf) ) {
        my $buffer = 0x00 x BUFFER_SIZE;
        my $gz = Compress::Zlib::gzopen( $dstf, 'wb' ); # wb => write binary
        while( read(IN, $buffer, BUFFER_SIZE) > 0 ) {
            $gz->gzwrite( $buffer );
        }
        $gz->gzflush(Z_FINISH);
        $gz->gzclose();
        close(IN);
        return 1;
    }
    return 0;
}

```

Ovviamente questo esempio è ridotto all'osso, ma se vi ha stuzzicato, andate pure a consultare la [documentazione <http://search.cpan.org/dist/Compress-Zlib>] completa.

--  
[cosimo]