zstd Fast compression

CONCEPTS

zstd keeps the input file by default and adds .zst. Use --rm to delete the original once it succeeds.

--fast=N fastest, largest
↓ -1
-3 default balance
↓ -19
--ultra -22 smallest, slow, lots of RAM
Level versus speed

COMPRESS

`zstd <file>`
Compress, keep original (<file>.zst)

`zstd <file> -o <out>`
Choose the output name

`zstd -<n> <file>`
Level 1-19 (default 3)

`zstd -c <file> > <out>`
Write to stdout, keep input

`zstd --rm <file>`
Remove input after success

`zstd -f <file>`
Overwrite existing output

`zstd -r <dir>`
Compress each file in a directory

`zstd <a> <b> <c>`
Compress several files at once

`zstd --fast=<n> <file>`
Faster than level 1, lower ratio

`zstd --ultra -22 <file>`
Levels 20-22, heavy memory use

DECOMPRESS

`zstd -d <file>.zst`
Decompress, keep the .zst

`unzstd <file>.zst`
Same as zstd -d

`zstd -dc <file>.zst`
Decompress to stdout

`zstdcat <file>.zst`
Print a compressed file

`zstdless <file>.zst`
Page a compressed file

DECOMPRESS (cont.)

`zstdgrep <pattern> <file>.zst`
Search inside a compressed file

`zstd -d <file>.zst -o <out>`
Choose the output name

`zstd --test <file>.zst`
Verify integrity only

INSPECT

`zstd -l <file>.zst`
Frames, sizes, ratio, checksum

SPEED AND MEMORY

`zstd -T0 <file>`
Use all physical cores

`zstd -T<n> <file>`
Use n threads

`zstd --long <file>`
Long-range matching for big, repetitive data

PIPES AND TAR

`tar --zstd -cf <a>.tar.zst <dir>`
Create a tar.zst

`tar --zstd -xf <a>.tar.zst`
Extract a tar.zst

`tar -tf <a>.tar.zst`
List contents

`tar -I 'zstd -T0 -19' -cf <a>.tar.zst <dir>`
Pick level and threads for tar

`tar -cf - <dir> | zstd -T0 > <a>.tar.zst`
Pipe tar into zstd

`zstd -dc <a>.tar.zst | tar -xf -`
Pipe zstd into tar

DICTIONARIES AND PATCHES

`zstd --train <files> -o <dict>`
Build a dictionary from samples

`zstd -D <dict> <file>`
Compress with a dictionary

`zstd -D <dict> -d <file>.zst`
Decompress with the same dictionary