

-m (Set compression Method) switch

Specifies the compression method.

Syntax

```
-m<method_parameters>
```

The format for this switch depends on the archive type.

- [Zip](#)
- [GZip](#)
- [BZip2](#)
- [7z](#)
- [XZ](#)
- [WIM](#)

-m switch also can specify hash method for [h \(Hash\)](#) command,

Notes: "Default value" in switches descriptions means the value that will be used if switch is not specified.

It's allowed to use reduced forms for boolean switches: **sw+** or **sw** instead **sw=on**, and **sw-** instead of **sw=off**.

Zip

Parameter	Default	Description
x=[0 1 3 5 7 9]	5	Sets level of compression.
m={MethodID}	Deflate	Sets a method: Copy, Deflate, Deflate64, BZip2, LZMA, PPMd.
fb={NumFastBytes}	32	Sets number of Fast Bytes for Deflate encoder.
pass={NumPasses}	1	Sets number of Passes for Deflate encoder.
d={Size}[b k m]	900000	Sets Dictionary size for BZip2
mem={Size}[b k m]	24	Sets size of used memory for PPMd.
o={Size}	8	Sets model order for PPMd.
mt=[off on {N}]	on	Sets multithreading mode.
em={EncryptionMethodID}	ZipCrypto	Sets a encryption method: ZipCrypto, AES128, AES192, AES256
tc=[off on]	on	Stores NTFS timestamps for files: Modification time, Creation time, Last access time.
cl=[off on]	off	7-Zip always uses local code page for file names.

cu=[off | on] off 7-Zip uses UTF-8 for file names that contain non-ASCII symbols.

cp={CodePage} off Sets code page

By default (if **cl** and **cu** switches are not specified), 7-Zip uses UTF-8 encoding only for file names that contain symbols unsupported by local code page.

x=[0 | 1 | 3 | 5 | 7 | 9]

Sets level of compression. x=0 means Copy mode (no compression).

Deflate / Deflate64 settings:

Level	NumFastBytes	NumPasses	Description
1			Fastest
3	32	1	Fast
5			Normal
7	64	3	Maximum
9	128	10	Ultra

x=1 and x=3 with Deflate method set fast mode for compression.

BZip2 settings:

Level	Dictionary	NumPasses	Description
1	100000		Fastest
3	500000	1	Fast
5			Normal
7	900000	2	Maximum
9		7	Ultra

fb={NumFastBytes}

Sets the number of fast bytes for the Deflate/Deflate64 encoder. It can be in the range from 3 to 258 (257 for Deflate64). Usually, a big number gives a little bit better compression ratio and a slower compression process. A large fast bytes parameter can significantly increase the compression ratio for files which contain long identical sequences of bytes.

pass={NumPasses}

Sets number of passes for Deflate encoder. It can be in the range from 1 to 15 for Deflate and from 1 to 10 for BZip2. Usually, a big number gives a little bit better compression ratio and a slower compression process.

d={Size}[b|k|m]

Sets the Dictionary size for BZip2. You must specify the size in bytes, kilobytes, or megabytes. The maximum value for the Dictionary size is 900000b. If you do not specify any symbol from set [b|k|m], dictionary size will be calculated as DictionarySize = 2^Size bytes.

mem={Size}[b|k|m]

Sets the size of memory used for PPMd. You must specify the size in bytes, kilobytes, or megabytes. The maximum value is 256 MB = 2^28 bytes. The default value is 24 (16MB). If you do not specify any symbol from the set [b|k|m], the

memory size will be calculated as (2^{Size}) bytes. PPMd uses the same amount of memory for compression and decompression.

o={Size}

Sets the model order for PPMd. The size must be in the range [2,16]. The default value is 8.

mt=[off | on | {N}]

Sets multithread mode. If you have a multiprocessor or multicore system, you can get a speed increase with this switch. This option affects only compression (with any method) and decompression of BZip2 streams. Each thread in the multithread mode uses 32 MB of RAM for buffering. If you specify {N}, 7-Zip tries to use N threads.

GZip

GZip uses the same parameters as Zip, but GZip compresses only with Deflate method. So GZip supports only the following parameters: x, fb, pass.

BZip2

Parameter	Default	Description
<code>x=[1 3 5 7 9]</code>	5	Sets level of compression.
<code>pass={NumPasses}</code>	1	Sets number of Passes for Bzip2 encoder.
<code>d={Size}[b k m]</code>	900000	Sets Dictionary size for BZip2
<code>mt=[off on {N}]</code>	on	Sets multithreading mode.

x=[1 | 3 | 5 | 7 | 9]

Sets level of compression

Level	Dictionary	NumPasses	Description
1	100000		Fastest
3	500000	1	Fast
5			Normal
7	900000	2	Maximum
9		7	Ultra

d={Size}[b|k|m]

Sets the Dictionary size for BZip2. You must specify the size in bytes, kilobytes, or megabytes. The maximum value for the Dictionary size is 900000b. If you do not specify any symbol from set [b|k|m], dictionary size will be calculated as $\text{DictionarySize} = 2^{\text{Size}}$ bytes.

pass={NumPasses}

Sets the number of passes. It can be in the range from 1 to 10. The default value is 1 for normal mode, 2 for maximum mode and 7 for ultra mode. A bigger number can give a little bit better compression ratio and a slower compression process.

mt=[off | on | {N}]

Sets multithread mode. If you have a multiprocessor or multicore system, you can get a speed increase with this switch. If you specify {N}, for example mt=4, 7-Zip tries to use 4 threads.

7z

Parameter	Default	Description
x=[0 1 3 5 7 9]	5	Sets level of compression.
yx=[0 1 3 5 7 9]	5	Sets level of file analysis.
s=[off on [e] [{N}f] [{N}b {N}k {N}m {N}g {N}t]	on	Sets solid mode.
qs=[off on]	off	Sort files by type in solid archives.
f=[off on FilterID]	on	Enables or disables filters. FilterID: Delta:{N}, BCJ, BCJ2, ARM, ARMT, IA64, PPC, SPARC.
hc=[off on]	on	Enables or disables archive header compressing.
he=[off on]	off	Enables or disables archive header encryption.
b{C1}[s{S1}]: {C2}[s{S2}]		Sets binding between coders.
{N}={MethodID} [:param1] [:param2][..]	LZMA2	Sets a method: LZMA, LZMA2, PPMd, BZip2, Deflate, Delta, BCJ, BCJ2, Copy.
mt=[off on {N}]	on	Sets multithreading mode.
mtf=[off on]	on	Set multithreading mode for filters.
tm=[off on]	on	Stores last Modified timestamps for files.
tc=[off on]	off	Stores Creation timestamps for files.
ta=[off on]	off	Stores last Access timestamps for files.
tr=[off on]	on	Stores file attributes.

x=[0 | 1 | 3 | 5 | 7 | 9]

Sets level of compression

Level	Method	Dictionary	FastBytes	MatchFinder	Filter	Description
0	Copy					No compression.
1	LZMA2	64 KB	32	HC4	BCJ	Fastest compressing
3	LZMA2	1 MB	32	HC4	BCJ	Fast compressing
5	LZMA2	16 MB	32	BT4	BCJ	Normal compressing

7	LZMA2	32 MB	64	BT4	BCJ	Maximum compressing
9	LZMA2	64 MB	64	BT4	BCJ2	Ultra compressing

Note: "x" works as "x=9".

yx=[0 | 1 | 3 | 5 | 7 | 9]

Sets level of file analysis

Level	Description
-------	-------------

0	No analysis.
1 or more	WAV file analysis (for Delta filter).
7 or more	EXE file analysis (for Executable filters).
9 or more	analysis of all files (Delta and executable filters).

Default level is 5: "yx=5".

"yx" works as "yx=9".

If the level of analysis is smaller than 9, 7-Zip analyses only files that have some file name extensions: EXE, DLL, WAV. 7-Zip reads small data block at the beginning of file and tries to parse the header. It supports only some formats: WAV, PE, ELF, Mach-O. Then it can select filter that can increase compression ratio for that file.

By default 7-Zip uses x86 filters (BCJ or BCJ2) for PE files (EXE, DLL). 7-Zip doesn't use analysis in default (yx=5) mode. If (yx=7), then analysis is used for PE files, and it can increase compression ratio for files for non-x86 platforms like ARM.

s=[off | on | [e] [{N}f] [{N}b | {N}k | {N}m | {N}g | {N}t)]

Enables or disables solid mode. The default mode is s=on. In solid mode, files are grouped together. Usually, compressing in solid mode improves the compression ratio.

e	Use a separate solid block for each new file extension. You need to use qs option also.
{N}f	Set the limit for number of files in one solid block
{N}b {N}k {N}m {N}g {N}t	Set a limit for the total size of a solid block in bytes / KiB / MiB / GiB / TiB.

These are the default limits for the solid block size:

Compression Level	Solid block size
Store	0 B
Fastest	16 MB
Fast	128 MB
Normal	2 GB
Maximum	4 GB
Ultra	4 GB

Limitation of the solid block size usually decreases compression ratio but gives the following advantages:

- Decreases losses in case of future archive damage.
- Decreases extraction time of a group of files (or just one file), so long as the group doesn't contain the entire archive.

The updating of solid .7z archives can be slow, since it can require some recompression.

Example:

```
s=100f10m
```

set solid mode with 100 files & 10 MB limits per one solid block.

qs=[off | on]

Enables or disables sorting files by type in solid archives. The default mode is qs=off.

Old versions of 7-Zip (before version 15.06) used file sorting "by type" ("by extension").

New versions of 7-Zip (starting from version 15.06) support two sorting orders:

- qs- : sorting by name : it's default order.
- qs : sorting by type (by file extension).

You can get big difference in compression ratio for different sorting methods, if dictionary size is smaller than total size of files. If there are similar files in different folders, the sorting "by type" can provide better compression ratio in some cases.

Note that sorting "by type" has some drawbacks. For example, NTFS volumes use sorting order "by name", so if an archive uses another sorting, then the speed of some operations for files with unusual order can fall on HDD devices (HDDs have low speed for "seek" operations).

If "qs" mode provides much better compression ratio than default "qs-" mode, you still can increase compression ratio for "qs-" mode by increasing of dictionary size.

If you think that unusual file order is not problem for you, and if better compression ratio with small dictionary is more important for you, use "qs" mode.

Note: There are some files (for example, executable files), that are compressed with additional filter. 7-Zip can't use different compression methods in one solid block, so 7-zip can create several groups of files that don't follow "by name" order in "qs-" mode, but files inside each group are still sorted by name in "qs-" mode.

f=[off | on | FilterID]

Enables or disables compression filters. The default mode is f=on, when 7-zip uses filter only for executable files: dll, exe, ocx, sfx, sys. It uses BCJ2 filter in Ultra mode and BCJ filter in other modes. If f=FilterID if specified, 7-zip uses specified filter for all files. FilterID can be: Delta:{N}, BCJ, BCJ2, ARM, ARMT, IA64, PPC, SPARC.

hc=[off | on]

Enables or disables archive header compressing. The default mode is hc=on. If archive header compressing is enabled, the archive header will be compressed with LZMA method.

he=[off | on]

Enables or disables archive header encryption. The default mode is he=off.

b{C1}[s{S1}]:{C2}[s{S2}]

Binds output stream S1 in coder C1 with input stream S2 in coder C2. If stream number is not specified, stream with number 0 will be used.

Usually coder has one input stream and one output stream. In 7z some coders can have multiple input and output streams.

For example, [BCJ2](#) encoder has one input stream and four output streams.

mt=[off | on | {N}]

Sets multithread mode. If you have a multiprocessor or multicore system, you can get a increase with this switch. 7-Zip supports multithread mode only for LZMA / LZMA2 compression and BZip2 compression / decompression. If you specify {N}, for example mt=4, 7-Zip tries to use 4 threads. LZMA compression uses only 2 threads.

{N}={MethodID}[:param1][:param2] ... [:paramN]

Sets compression method. You can use any number of methods. The default method is LZMA2.

{N} sets the index number of method in methods chain. Numbers must begin from 0. Methods that have smaller numbers will be used before others.

Parameters must be in one of the following forms:

- {ParamName}={ParamValue}.
- {ParamName}{ParamValue}, if {ParamValue} is number and {ParamName} doesn't contain numbers.

Supported methods:

MethodID	Description
LZMA	LZ-based algorithm
LZMA2	LZMA-based algorithm
PPMd	Dmitry Shkarin's PPMdH with small changes
BZip2	BWT algorithm
Deflate	LZ+Huffman
Copy	No compression

Supported filters:

MethodID	Description
Delta	Delta filter
BCJ	converter for x86 executables
BCJ2	converter for x86 executables (version 2)
ARM	converter for ARM (little endian) executables
ARMT	converter for ARM Thumb (little endian) executables
IA64	converter for IA-64 executables
PPC	converter for PowerPC (big endian) executables
SPARC	converter for SPARC executables

Filters increase the compression ratio for some types of files. Filters must be used

with one of the compression method (for example, BCJ + LZMA).

LZMA

LZMA is an algorithm based on Lempel-Ziv algorithm. It provides very fast decompression (about 10-20 times faster than compression). Memory requirements for compression and decompression also are different (see [d={Size}\[b|k|m|g\]](#) switch for details).

Parameter	Default	Description
a=[0 1]	1	Sets compressing mode
d={Size}[b k m g]	24	Sets Dictionary size
mf={MF_ID}	bt4	Sets Match Finder
fb={N}	32	Sets number of Fast Bytes
mc={N}	32	Sets Number of Cycles for Match Finder
lc={N}	3	Sets number of Literal Context bits - [0, 8]
lp={N}	0	Sets number of Literal Pos bits - [0, 4]
pb={N}	2	Set number of Pos Bits - [0, 4]

[a=\[0|1\]](#)

Sets compression mode: 0 = fast, 1 = normal. Default value is 1.

[d={Size}\[b|k|m|g\]](#)

Sets Dictionary size for LZMA. You must specify the size in bytes, kilobytes, or megabytes. The maximum value for dictionary size is 1536 MB, but 32-bit version of 7-Zip allows to specify up to 128 MB dictionary. Default values for LZMA are 24 (16 MB) in normal mode, 25 (32 MB) in maximum mode (-mx=7) and 26 (64 MB) in ultra mode (-mx=9). If you do not specify any symbol from the set [b|k|m|g], the dictionary size will be calculated as DictionarySize = 2^{Size} bytes. For decompressing a file compressed by LZMA method with dictionary size N, you need about N bytes of memory (RAM) available.

[mf={MF_ID}](#)

Sets Match Finder for LZMA. Default method is bt4. Algorithms from hc* group don't provide a good compression ratio, but they often work pretty fast in combination with fast mode (a=0). Memory requirements depend on dictionary size (parameter "d" in table below).

MF_ID	Dictionary	Memory Usage	Description
bt2		9.5 * d	2 bytes hashing
bt3		11.5 * d	3 bytes hashing
bt4	64 KB ... 48 MB	11.5 * d	Binary Tree
	64 MB ... 1024 MB	10.5 * d	
hc4	64 KB ... 48 MB	7.5 * d	Hash Chain
	64 MB ... 1024 MB	6.5 * d	

Note: Your operation system also needs some amount of physical memory for internal purposes. So keep at least 32MB of physical memory unused.

fb={N}

Sets number of fast bytes for LZMA. It can be in the range from 5 to 273. The default value is 32 for normal mode and 64 for maximum and ultra modes. Usually, a big number gives a little bit better compression ratio and slower compression process.

mc={N}

Sets number of cycles (passes) for match finder. It can be in range from 0 to 1000000000. Default value is $(16 + \text{number_of_fast_bytes} / 2)$ for BT* match finders and $(8 + \text{number_of_fast_bytes} / 4)$ for HC4 match finder. If you specify mc=0, LZMA will use default value. Usually, a big number gives a little bit better compression ratio and slower compression process. For example, mf=HC4 and mc=10000 can provide almost the same compression ratio as mf=BT4.

lc={N}

Sets the number of literal context bits (high bits of previous literal). It can be in range from 0 to 8. Default value is 3. Sometimes lc=4 gives gain for big files.

lp={N}

Sets the number of literal pos bits (low bits of current position for literals). It can be in the range from 0 to 4. The default value is 0. The lp switch is intended for periodical data when the period is equal to 2^{value} (where lp=value). For example, for 32-bit (4 bytes) periodical data you can use lp=2. Often it's better to set lc=0, if you change lp switch.

pb={N}

Sets the number of pos bits (low bits of current position). It can be in the range from 0 to 4. The default value is 2. The pb switch is intended for periodical data when the period is equal 2^{value} (where lp=value).

LZMA2

LZMA2 is modified version of LZMA. it provides the following advantages over LZMA:

- Better compression ratio for data than can't be compressed. LZMA2 can store such blocks of data in uncompressed form. Also it decompresses such data faster.
- Better multithreading support. If you compress big file, LZMA2 can split that file to chunks and compress these chunks in multiple threads.

Parameter	Default	Description
c={Size} [b k m g]	dictSize * 4	Sets Chunk size

If you don't specify ChunkSize, LZMA2 sets it to $\max(\text{DictionarySize}, \min(256\text{M}, \max(1\text{M}, \text{DictionarySize} * 4)))$.

LZMA2 also supports all LZMA parameters, but lp+lc cannot be larger than 4.

LZMA2 uses: 1 thread for each chunk in x1 and x3 modes; and 2 threads for each chunk in x5, x7 and x9 modes. If LZMA2 is set to use only such number of threads required for one chunk, it doesn't split stream to chunks. So you can get different compression ratio for different number of threads. You can get the best compression ratio, when you use 1 or 2 threads.

PPMd

PPMd is a PPM-based algorithm. This algorithm is mostly based on Dmitry Shkarin's PPMdH source code. PPMd provides very good compression ratio for plain text files. There is no difference between compression speed and decompression speed. Memory requirements for compression and decompression also are the same.

Parameter	Default	Description
<code>mem={Size}[b k m g]</code>	24	Sets size of used memory for PPMd.
<code>o={Size}</code>	6	Sets model order for PPMd.

mem={Size}[b|k|m|g]

Sets the size of memory used for PPMd. You must specify the size in bytes, kilobytes, or megabytes. The maximum value is 2GB = 2^{31} bytes. The default value is 24 (16MB). If you do not specify any symbol from the set [b|k|m|g], the memory size will be calculated as (2^{Size}) bytes. PPMd uses the same amount of memory for compression and decompression.

o={Size}

Sets the model order for PPMd. The size must be in the range [2,32]. The default value is 6.

BCJ2

BCJ2 is a Branch converter for 32-bit x86 executables (version 2). It converts some branch instructions for increasing further compression.

A BCJ2 encoder has one input stream and four output streams:

- s0: main stream. It requires further compression.
- s1: stream for converted CALL values. It requires further compression.
- s2: stream for converted JUMP values. It requires further compression.
- s3: service stream. It is already compressed.

If LZMA is used, the size of the dictionary for streams s1 and s2 can be much smaller (512 KB is enough for most cases) than the dictionary size for stream s0.

Parameters:

d={Size}[b|k|m|g]

Sets section size for BCJ2 filter. Default section size is 64 MB. If you do not specify any symbol from the set [b|k|m|g], the section size will be calculated as $\text{SectionSize} = 2^{\text{Size}}$ bytes. This parameter doesn't affect memory consumption. Compression ratio is better, if the section size is equal or slightly larger than size of largest execution section in file. Example: `f=BCJ2:d9M`, if largest executable section in files is smaller than 9 MB.

Delta

It's possible to set delta offset in bytes. For example, to compress 16-bit stereo WAV files, you can set `"0=Delta:4"`. Default delta offset is 1.

XZ

XZ supports only LZMA2 codec now. The switches are similar to switches for 7z format.

Parameter	Default	Description
x=[1 3 5 7 9]	5	Sets level of compression
f=FilterID		Sets compression filter. FilterID: Delta:{N}, BCJ, ARM, ARMT, IA64, PPC, SPARC
{N}={MethodID} [:param1] [:param2][..]	LZMA2	Sets compression method: LZMA2:[param1]:[param2]:[...]
mt=[off on {N}]	on	Sets multithreading mode
s=[off on {N}b {N}k {N}m {N}g]	off	Sets solid mode.

s=[off | on | [{N}b | {N}k | {N}m | {N}g]]

Enables or disables solid mode. The default mode is s=off. In solid mode, there is only one block per file or stream.

{N}b | {N}k | {N}m | {N}g Set a limit for the total size of a solid block in bytes

If size of solid block is not specified, default value of solid block size will be calculated, that depends from "compression level" and "dictionary size":

dictionary_size	Default solid block size
smaller than 256 KB	1 MB
256 KB - 64 MB	dictionary_size * 4
64 MB - 256 MB	256 MB
larger than 256 MB	dictionary_size

block size must be equal or large than dictionary size.

If you use multiple blocks:

- the compression ratio with small blocks usually is worse.
- blocks are independent. So losses in case of data damage is limited only to damaged blocks.
- it's possible to extract some particular block of data faster.
- there is index record at the end of xz stream that contains information about position and size of each block.

Note: xz uses: 1 thread for each block in x1 and x3 modes; and 2 threads for each block in x5, x7 and x9 modes. If xz is set to use only such number of threads required for one block, it doesn't split stream to blocks. So you can get different compression ratio for different number of threads. You can get the best compression ratio, when you use 1 thread (for x1 and x3 modes) or 2 threads (for x5, x7 and x9 modes).

Note: each xz block contains LZMA2 stream of data. And LZMA2 also can be divided to independent blocks (chunks). The difference between xz blocks and LZMA2 blocks, that each xz block contains also checksum (crc or sha), and there is index

record at the end of xz stream that points to each xz block. 7-Zip by default uses xz blocks. But it's possible to specify the mode when it will use one xz block, and multiple LZMA2 blocks instead.

Examples:

```
s=16m
```

use 16 MB blocks.

```
s
```

use one solid xz block per file.

```
s 0c16m
```

use one solid xz block per file and 16 MiB LZMA2 blocks.

WIM

Parameter	Default	Description
<code>im={ImageNumber}</code>		Sets image number.
<code>is=[off on]</code>	off	Show image number in paths.

If image number is specified, 7-Zip works only with that image inside WIM archive. Other images will be not changed. By default 7-Zip doesn't show image number, if there is only one image in WIM archive, or if image number is specified. But if the switch "is" specified, 7-Zip shows image number.

Examples

```
7z a archive.zip *.jpg -mx0
```

adds `*.jpg` files to `archive.zip` archive without compression.

```
7z a archive.7z *.exe *.dll -m0=BCJ -m1=LZMA:d=21
```

adds `*.exe` and `*.dll` files to solid archive `archive.7z` using LZMA method with 2 MB dictionary and BCJ filter.

```
7z a archive.7z a.tar -mf=BCJ2 -mx
```

adds `a.tar` files to archive `archive.7z` using BCJ2 filter.

```
7z a archive.7z *.wav -mf=Delta:4
```

adds *.wav files to archive archive.7z using Delta:4 filter.

```
7z a a.7z *.exe *.dll -m0=BCJ2 -m1=LZMA:d25 -m2=LZMA:d19  
-m3=LZMA:d19 -mb0:1 -mb0s1:2 -mb0s2:3
```

adds *.exe and *.dll files to archive a.7z using BCJ2 filter, LZMA with 32 MB dictionary for main output stream (s0), and LZMA with 512 KB dictionary for s1 and s2 output streams of BCJ2.

```
7z a archive.7z *.txt -m0=PPMd
```

adds *.txt files to archive archive.7z using PPMd method.

```
7z a a.tar.xz a.tar -mf=bcj -mx
```

adds a.tar files to archive a.tar.xz using BCJ filter.

Commands that can be used with this switch

[a](#) (Add), [h](#) (Hash), [d](#) (Delete), [rn](#) (Rename), [u](#) (Update)

See also

Switches: [-t](#) (set Type of archive),