7Zip

Syntax

Command Line Syntax

7z <command> [<switch>...] <base_archive_name> [<arguments>...]
<arguments> ::= <switch> | <wildcard> | <filename> | [@listfile]
<switch>::= -{switch_name}
Expressions in square brackets (between '[' and ']') are optional.
Expressions in curly braces ('{' and '}') mean that instead of that Expression (including braces), the user must substitute some string.
Expression
expression1 | expression2 | ... | expressionN
means that any (but only one) from these expressions must be specified.
Commands and switches can be entered in upper or lower case.
Command is the first non-switch argument.
The "base_archive_name" must be the first filename on the command line after the command.
The switches and other filenames can be in any order.
Wildcards or filenames with spaces must be quoted:

```
"Dir\Program files\*"
Dir\"Program files"\*
```

Switch options can be combined to save command line length. However, some switch options take optional string arguments and therefore, must be the last option in a combined argument token string because 7-Zip accepts the rest of the argument token as the optional argument.
7-Zip uses wild name matching similar to Windows 95:

- **'*'** means a sequence of arbitrary characters.
- **'?'** means any character.

7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.
Examples:

click me	**click me**
*.txt	means all files with an extension of ".txt"
?a*	means all files with a second character of "a"
1	means all names that contains character "1"
..*	means all names that contain two at least "." characters

The default wildcard "*" will be used if there is no filename/wildcard in the command line.
Slash ('\') at the end of a path means a directory. Without a Slash ('\') at the end of the path, the path can refer either to a file or a directory.

List file

You can supply one or more filenames or wildcards for special list files (files containing lists of files). The filenames in such list file must be separated by new line symbol(s).
For list files, 7-Zip uses UTF-8 encoding by default. You can change encoding using -scs switch.
Multiple list files are supported.
For example, if the file "listfile.txt" contains the following:

```
My programs\*.cpp
Src\*.cpp
```

then the command

```
7z a -tzip archive.zip @listfile.txt
```

adds to the archive "archive.zip" all "*.cpp" files from directories "My programs" and "Src".

Short and Long File Names

7-Zip supports short file names (like FILENA~1.TXT) in some cases. However, it's strongly recommended to use only the real (long) file names.

Commands Line

a (Add)

a (Add) command
Adds files to archive.

Examples
7z a archive1.zip subdir\
adds all files and subfolders from folder subdir to archive archive1.zip. The filenames in archive will contain subdir\ prefix.
7z a archive2.zip .\subdir*
adds all files and subfolders from folder subdir to archive archive2.zip. The filenames in archive will not contain subdir\ prefix.
cd /D c:\dir1\
7z a c:\archive3.zip dir2\dir3\
The filenames in archive c:\archive3.zip will contain dir2\dir3\ prefix, but they will not contain c:\dir1\ prefix.7z a Files.7z *.txt -r
adds all *.txt files from current folder and its subfolders to archive Files.7z.

Notes
7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.

Switches that can be used with this command
-i (Include)
-m (Method)
-p (Set Password)
-r (Recurse)
-sdel (Delete files after including to archive)
-sfx (create SFX)
-si (use StdIn)
-sni (Store NT security information)
-sns (Store NTFS alternate Streams)
-so (use StdOut)
-spf (Use fully qualified file paths)
-ssw (Compress shared files)
-stl (Set archive timestamp from the most recently modified file)
-t (Type of archive)
-u (Update)
-v (Volumes)
-w (Working Dir)
-x (Exclude)

b (Benchmark)

b (Benchmark) command
Measures speed of the CPU.
Benchmark execution also can be used to check RAM for errors.

Syntax
b [number_of_iterations] [-mmt{N}] [-md{N}] [-mm={Method}]
The LZMA benchmark is default benchmark for benchmark command.
There are two tests for LZMA benchmark:

1. Compressing with LZMA method
2. Decompressing with LZMA method

The LZMA benchmark shows a rating in MIPS (million instructions per second). The rating value is calculated from the measured speed, and it is normalized with results of Intel Core 2 CPU with multi-threading option switched off. So if you have modern CPU from Intel or AMD, rating values in single-thread mode must be close to real CPU frequency.
You can change the upper dictionary size to increase memory usage by -md{N} switch. Also, you can change the number of threads by -mmt{N} switch.
The **Dict** column shows dictionary size. For example, 21 means 2^21 = 2 MB.
The **Usage** column shows the percentage of time the processor is working. It's normalized for a one-thread load. For example, 180% CPU Usage for 2 threads can mean that average CPU usage is about 90% for each thread.
The **R / U** column shows the rating normalized for 100% of CPU usage. That column shows the performance of one average CPU thread.
Avr shows averages for different dictionary sizes.

Tot shows averages of the compression and decompression ratings.
The test data that is used for compression in that test is produced with special algorithm, that creates data stream that has some properties of real data, like text or execution code. Note that the speed of LZMA for real data can be slightly different.

LZMA benchmark details

Compression speed strongly depends from memory (RAM) latency, Data Cache size/speed and TLB. Out-of-Order execution feature of CPU is also important for that test.
Decompression speed strongly depends on CPU integer operations. The most important things for that test are: branch misprediction penalty (the length of pipeline) and the latencies of 32-bit instructions ("multiply", "shift", "add" and other). The decompression test has very high number of unpredictable branches. Note that some CPU architectures (for example, 32-bit ARM) support instructions that can be conditionally executed. So such CPUs can work without branches (and without pipeline flushing) in many cases in LZMA decompression code. And such CPUs can have some speed advantages over other architectures that don't support complex conditionally execution. Out-of-Order execution capability is not so important for LZMA Decompression.
The test code doesn't use FPU and SSE. Most of the code is 32-bit integer code. Only some minor part in compression code uses also 64-bit integers. RAM and Cache bandwidth are not so important for these tests. The latencies are much more important.
The CPU's IPC (Instructions per cycle) rate is not very high for these tests. The estimated value of test's IPC is 1 (one instruction per cycle) for modern CPU. The compression test has big number of random accesses to RAM and Data Cache. So big part of execution time the CPU waits the data from Data Cache or from RAM. The decompression test has big number of pipeline flushes after mispredicted branches. Such low IPC means that there are some unloaded CPU resources. But the CPU with Hyper-Threading feature can load these CPU resources using two threads. So Hyper-Threading provides pretty big improvement in these tests.

LZMA benchmark in multithreading mode

When you specify (N*2) threads for test, the program creates N copies of LZMA encoder, and each LZMA encoder instance compresses separated block of test data. Each LZMA encoder instance creates 3 unsymmetrical execution threads: two big threads and one small thread. The total CPU load for these 3 threads can vary from 140% to 200%. To provide better CPU load during compression, you can test the mode, where the number of benchmark threads is larger than the number of hardware threads.
Each LZMA encoder instance in multithreading mode divides the task of compression into 3 different tasks, where each task is executed in separated thread. Each of these tasks is simpler than original task, and it uses less memory. So each thread uses the data cache and TLB more effectively in multithreading mode. And LZMA encoder is slightly more effective in multithreading mode in value of "the Speed" divided to "CPU usage".
Note that there is some data traffic between 3 threads of LZMA encoder. So data exchange bandwidth via memory between CPU threads is also can be important, especially in multi-core system with big number of cores or CPUs.
All LZMA decoder threads are symmetrical and independent. So the decompression test uses all hardware threads, if the number of hardware threads is used.

7-Zip benchmark

With -mm=* switch you can run a complex benchmark for 7-Zip code. It tests hash calculation methods, compression and encryption codecs of 7-Zip. Note that the tests of LZMA have big weight in "total" results. And the results are normalized with AMD K8 cpu in that complex benchmark.
The **CPU** rows show CPU frequency. It's measured for sequence of simple CPU instructions. Note: It can be inaccurate, if hyper-threading is used.
The **Effec** column shows Efficiency - the Rating normalized to CPU frequency.
The **E / U** column shows the Efficiency normalized for 100% of CPU usage.

Examples

7z b
runs benchmarking.7z b -mmt1 -md26
runs benchmarking with one thread and 64 MB dictionary.7z b 30
runs benchmarking for 30 iterations. It can be used to check RAM for errors.
7z b -mm=*
runs complex 7-Zip benchmark.
7z b -mm=* -mmt=*
runs complex 7-Zip benchmark for different number of threads : (1, max/2, max), where max is number of available hardware threads. So it can test 3 main modes: single-thread, multi-thread without hyper-threading, multi-thread with hyper-threading.

d (Delete)

d (Delete) command

Deletes files from archive.

Example

7z d archive.zip *.bak -r
deletes *.bak files from archive archive.zip.

Notes
7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.

Switches that can be used with this command
-i (Include)
-m (Method)
-p (Set Password)
-r (Recurse)
-sns (Store NTFS alternate Streams)
-u (Update)
-w (Working Dir)
-x (Exclude)

e (Extract)

e (Extract) command
Extracts files from an archive to the current directory or to the output directory. The output directory can be specified by -o (Set Output Directory) switch.
This command copies all extracted files to one directory. If you want extract files with full paths, you must use x (Extract with full paths) command.
7-Zip will prompt the user before overwriting existing files unless the user specifies the -y (Assume Yes on all queries) switch. If the user gives a **no** answer, 7-Zip will prompt for the file to be extracted to a new filename. Then a **no** answer skips that file; or, **yes** prompts for new filename.
7-Zip accepts the following responses:

Answer	**Abbr.**	**Action**
Yes	y	
No	n	
Always	a	Assume YES for ALL subsequent queries of the same class
Skip	s	Assume NO for ALL subsequent queries of the same class
Quit	q	Quit the program

Abbreviated responses are allowed.

Examples
7z e archive.zip
extracts all files from archive archive.zip to the current directory.
7z e archive.zip -oc:\soft *.cpp -r
extracts all *.cpp files from archive archive.zip to c:\soft folder.

Notes
7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.

Switches that can be used with this command
-ai (Include archives)
-an (Disable parsing of archive_name)
-ao (Overwrite mode)
-ax (Exclude archives)
-i (Include)
-m (Method)
-o (Set Output Directory)
-p (Set Password)
-r (Recurse)
-si (use StdIn)
-sni (Store NT security information)
-sns (Store NTFS alternate Streams)
-so (use StdOut)
-spf (Use fully qualified file paths)
-t (Type of archive)
-x (Exclude)
-y (Assume Yes on all queries)

h (Hash)

h (Hash) command
Calculate hash values for files.

Syntax
h [-scrc{Method}] [files]
Supported methods: CRC32, CRC64, SHA1, SHA256, BLAKE2sp. Default method is CRC32.

Examples
7z h a.txt
calculates CRC32 for a.txt.
7z h -scrcsha256 a.iso
calculates SHA256 for a.iso.
7z h *
calculates CRC32 for all files in current folder and all subfolders.

Notes
7-Zip shows hash values for each file, the sum of hash values and the sum that includes all hash values of data and all hash values for filenames.
7-Zip represents hash values for CRC32 and CRC64 as integer numbers in hex.
7-Zip represents hash values For SHA1, SHA256 and BLAKE2sp as sequence of bytes in hex.

Switches that can be used with this command
-i (Include)
-m (Method)
-r (Recurse)
-scrc (Set hash method)
-si (use StdIn)
-sns (Store NTFS alternate Streams)
-ssw (Compress shared files)
-x (Exclude)

i (Information)

Show information about supported formats

l (List)

l (List contents of archive) command
Lists contents of archive.

Examples
7z l archive.zip
lists all files from archive archive.zip.

Notes
7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.

Switches that can be used with this command
-ai (Include archives)
-an (Disable parsing of archive_name)
-ax (Exclude archives)
-i (Include)
-slt (Show technical information)
-sns (Store NTFS alternate Streams)
-p (Set Password)
-r (Recurse)
-t (Type of archive)

-x (Exclude)

rn (Rename)

rn (Rename) command
Renames files in archive.

Syntax
rn <archive_name> <src_file_1> <dest_file_1> [<src_file_2> <dest_file_2> ...]

Example
7z rn a.7z old.txt new.txt 2.txt folder\2new.txt
renames old.txt to new.txt and 2.txt to folder\2new.txt .

Notes

Switches that can be used with this command
-i (Include)
-m (Method)
-p (Set Password)
-r (Recurse)
-stl (Set archive timestamp from the most recently modified file)
-u (Update)
-w (Working Dir)
-x (Exclude)

t (Test)

t (Test integrity of archive) command
Tests archive files.

Example
7z t archive.zip *.doc -r
tests *.doc files in archive archive.zip.

Notes
7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.

Switches that can be used with this command
-ai (Include archives)
-an (Disable parsing of archive_name)
-ax (Exclude archives)
-i (Include)
-p (Set Password)
-r (Recurse)
-sns (Store NTFS alternate Streams)
-x (Exclude)

u (Update)

u (Update) command
Update older files in the archive and add files that are not already in the archive.
Note: the updating of solid .7z archives can be slow, since it can require some recompression.

Example
7z u archive.zip *.doc
updates *.doc files to archive archive.zip.

Notes
7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.

Switches that can be used with this command
-i (Include)
-m (Method)
-p (Set Password)
-r (Recurse)
-sfx (create SFX)
-si (use StdIn)
-so (use StdOut)
-sni (Store NT security information)
-sns (Store NTFS alternate Streams)
-ssw (Compress shared files)
-spf (Use fully qualified file paths)
-stl (Set archive timestamp from the most recently modified file)
-t (Type of archive)
-u (Update)
-w (Working Dir)
-x (Exclude)

x (Extract)

x (Extract with full paths) command
Extracts files from an archive with their full paths in the current directory, or in an output directory if specified.
See the e (Extract) command description for more details.

Examples
7z x archive.zip
extracts all files from the archive archive.zip to the current directory.
7z x archive.zip -oc:\soft *.cpp -r
extracts all *.cpp files from the archive archive.zip to c:\soft folder.

Notes
7-Zip doesn't use the system wildcard parser. 7-Zip doesn't follow the archaic rule by which *.* means any file. 7-Zip treats *.* as matching the name of any file that has an extension. To process all files, you must use a * wildcard.

Switches that can be used with this command
-ai (Include archives)
-an (Disable parsing of archive_name)
-ao (Overwrite mode)
-ax (Exclude archives)
-i (Include)
-m (Method)
-o (Set Output Directory)
-p (Set Password)
-r (Recurse)
-si (use StdIn)
-sni (Store NT security information)
-sns (Store NTFS alternate Streams)
-so (use StdOut)
-spf (Use fully qualified file paths)
-t (Type of archive)
-x (Exclude)
-y (Assume Yes on all queries)

Commands Arguments

-i (Include)

-i (Include filenames) switch
Specifies additional include filenames and wildcards.
Multiple include switches are supported.

Syntax
-i[<recurse_type>]<file_ref>

<recurse_type> ::= r[- | 0]
<file_ref> ::= @{listfile} | !{wildcard}

Parameters

<recurse_type>
Specifies how wildcards and file names in this switch must be used. If this option is not given, then the global value, assigned by the -r (Recurse) switch will be used. For more details see specification of the -r (Recurse) switch.
<recurse_type> ::= r[- | 0]

<file_ref>
Specifies filenames and wildcards, or a list file, for files to be processed.
<file_ref> ::= @{listfile} | !{wildcard}

Option	Description
{listfile}	Specifies name of list file. See List file description.
{wildcard}	Specifies wildcard or filename.

Examples
7z a -tzip src.zip *.txt -ir!DIR1*.cpp
adds to src.zip archive all *.txt files from current directory and all *.cpp files from directory DIR1 and from all it's subdirectories.

Commands that can be used with this switch
a (Add), d (Delete), e (Extract), h (Hash), l (List), rn (Rename), t (Test), u (Update), x (Extract with full paths)

-m (Method)

-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	32	1	Fastest
3			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	1	Fastest
3	500000		Fast
5	900000		Normal
7		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
x=[1 | 3 | 5 | 7 | 9]	5	Sets level of compression.
pass={NumPasses}	1	Sets number of Passes for Bzip2 encoder.
d={Size}[b|k|m]	900000	Sets Dictionary size for BZip2
mt=[off | on | {N}]	on	Sets multithreading mode.

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

Sets level of compression

Level	Dictionary	NumPasses	Description
1	100000	1	Fastest
3	500000		Fast
5	900000		Normal
7		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 DictionarySize = 2^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
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
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.

click me	click me
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.
Usally 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).

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 + number_of_fast_bytes / 2) for BT* match finders and (8 + 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(DictionarySize, min(256M, max(1M, 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
mem={Size}[b|k|m|g]	24	Sets size of used memory for PPMd.
o={Size}	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 SectionSize = 2^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.

click me	click me
{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
im={ImageNumber}		Sets image number.
is=[off | on]	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)

-p (Password)

-p (set Password) switch

Specifies password.

Syntax

-p{password}

{password}
Specifies password.

Examples

7z a archive.7z -psecret -mhe *.txt
compresses *.txt files to archive.7z using password "secret". Also it encrypts archive headers (-mhe switch), so filenames will be

encrypted.
7z x archive.zip -psecret
extracts all files from archive.zip using password "secret".

Commands that can be used with this switch
a (Add), d (Delete), e (Extract), rn (Rename), t (Test), u (Update), x (Extract with full paths)

-r (Recurse)

-r (Recurse subdirectories) switch
Specifies the method of treating wildcards and filenames on the command line.

Syntax
-r[- | 0]

Switch	Description
-r	Enable recurse subdirectories.
-r-	Disable recurse subdirectories. This option is default for all commands.
-r0	Enable recurse subdirectories only for wildcard names.

Examples
7z l archive.zip *.doc -r-
lists all *.doc files that belong to the archived root directory in the archive.zip archive.
7z a -tzip archive.zip -r src*.cpp src*.h
adds all *.cpp and *.h files from directory src and all it's subdirectories to the archive.zip archive.
7z a archive.7z folder1\
adds all files from directory folder1 and all it's subdirectories to the archive.7z archive.
7z a archive.7z -r folder2\
searches all folder2 directories in all subdirectories, and adds them (including all subdirectories) to the archive.7z archive.

Commands that can be used with this switch
a (Add), d (Delete), e (Extract), h (Hash), l (List), rn (Rename), t (Test), u (Update), x (Extract with full paths)

-sdel (Delete)

-sdel (Delete files after compression) switch

Syntax
-sdel
If -sdel switch is specified, 7-Zip deletes files after including to archive. So it works like moving files to archive.
7-Zip deletes files at the end of operation and only if archive was successfully created.

Examples
7z a a.7z *.txt -sdel
moves txt files from disk's directory to a.7z archive.

Commands that can be used with this switch
a (Add)

-sfx (SFX) file

-sfx (Create SFX archive) switch
Creates self extracting archive.

Syntax
-sfx[{SFX_Module}]

{SFX_Module}

Specifies the SFX module that will be combined with the archive. This module must be placed in the same directory as the 7z.exe. If {SFX_Module} is not assigned, 7-Zip will use standard console SFX module 7zCon.sfx.

SFX_Module	Description
7z.sfx	SFX module (GUI version)
7zCon.sfx	SFX module (Console version)
7zSD.sfx	SFX module for installers (GUI version)
7zS2.sfx	small SFX module for installers (GUI version)
7zS2con.sfx	small SFX module for installers (Console version)

SFX module can unpack 7z archive or 7z multivolume archive. For example, if you have name.7z or name.7z.001 archive, just rename sfx muldule to name.exe and place to same folder with archive.

SFX modules for installers

SFX modules for installers are included in an external package (LZMA SDK). You can download these modules from www.7-zip.org. SFX module for installers (7zSD.sfx) allow you to create your own installation program. Such a module extracts the archive to the user's temp folder, and runs a specified program, and removes the temp files after the program finishes. A self-extracting archive for installers must be created as joining the following files: SFX_Module, Installer_Config (optional), 7z_Archive. You can use the following command to create an installer self-extracting archive:

copy /b 7zSD.sfx + config.txt + archive.7z archive.exe

An optimally small installation package size can be achieved, if the installation files are uncompressed before including them in the 7z archive.

-y switch for installer module specifies quiet mode extraction.

Installer Config file format

This config file contains commands for the Installer. The file begins with the string **;!@Install@!UTF-8!** and ends with **;!@InstallEnd@!**. The file must be written in UTF-8 encoding. The file contains any or all these string pairs:

ID_String="Value"

ID_String	Description
Title	Title for messages
BeginPrompt	Begin Prompt message
Progress	Value can be "yes" or "no". Default value is "yes".
RunProgram	Command for executing. Default value is "setup.exe". Substring %%T will be replaced with path to temporary folder, where files were extracted
Directory	Directory prefix for "RunProgram". Default value is ".\\"
ExecuteFile	Name of file for executing
ExecuteParameters	Parameters for "ExecuteFile"

You may omit any pair.

There are two ways to run a installation program: **RunProgram** and **ExecuteFile**. Use **RunProgram**, if you want to run a program from the .7z archive. Use **ExecuteFile**, if you want to open a document from the .7z archive, or if you want to execute a command from Windows.

If you use **RunProgram**, and if you specify empty directory prefix: **Directory**="", the system searches for the executable file in the following sequence:

1. The directory from which the application (installer) loaded.
2. The temporary folder, where files were extracted.
3. The Windows system directory.

Config file Examples

```
;!@Install@!UTF-8!
Title="7-Zip 4.00"
BeginPrompt="Do you want to install the 7-Zip 4.00?"
RunProgram="setup.exe"
;!@InstallEnd@!

;!@Install@!UTF-8!
Title="7-Zip 4.00"
BeginPrompt="Do you want to install the 7-Zip 4.00?"
ExecuteFile="7zip.msi"
;!@InstallEnd@!

;!@Install@!UTF-8!
Title="7-Zip 4.01 Update"
```

```
BeginPrompt="Do you want to install the 7-Zip 4.01 Update?"
ExecuteFile="msiexec.exe"
ExecuteParameters="/i 7zip.msi REINSTALL=ALL REINSTALLMODE=vomus"
;!@InstallEnd@!
```

Examples

7z a -sfx a.exe *.txt
adds *.txt files to self extracting archive a.exe using the default console SFX module.
7z a -sfx7z.sfx a.exe *
adds all files to self extracting archive a.exe with module 7z.sfx using windows version of SFX mudule.

Commands that can be used with this switch

a (Add), d (Delete), u (Update),

-si (STDIN)

-si (read data from stdin) switch

Causes 7-Zip to read data from stdin (standard input) instead of from disc files.

Syntax

-si{file_name}

{file_name}
Specifies a name that will be stored in the archive for the compressed data. If file_name is not specified, data will be stored without a name.Note: The current version of 7-Zip support reading of archives from stdin only for xz, lzma, tar, gzip and bzip2 archives.

Examples

7z a archive.gz -tgzip -siDoc2.txt < Doc.txt
compresses input stream from file Doc.txt to archive.gz archive using Doc2.txt file name.
7z x 7z905.tar.gz -so | 7z x -si -ttar
decompresses tar.gz archive.

Commands that can be used with this switch

a (Add), e (Extract), h (Hash), u (Update), x (Extract with full paths)

-so (STDOUT)

-so (write data to stdout) switch

Causes 7-Zip to write output data to stdout (standard output stream).

Syntax

-so
If the -so switch is used with the command that creates archive, it works only with some archive formats: xz, gzip, bzip2 and tar.

Examples

7z x archive.gz -so > Doc.txt
decompresses archive.gz archive to output stream and then redirects that stream to Doc.txt file.
7z a dummy -tgzip -so Doc.txt > archive.gz
compresses the Doc.txt file to the 7-Zip standard output stream and writes that stream to archive.gz file.

Commands that can be used with this switch

a (Add), e (Extract), u (Update), x (Extract with full paths)

-sni (NTFS)

-sni (Store NT security information) switch

Syntax
-sni
Use this switch to store and restore NT (NTFS) security information for files and directories. Note that only NTFS file system supports that feature.
Current version of 7-Zip can store NT security information only to WIM archives.

Examples
7z a a.wim -sni *.txt
stores txt files with NT security information.
7z x a.wim -sni
unpacks a.wim and restores NT security information.

Commands that can be used with this switch
a (Add), e (Extract), u (Update), x (Extract with full paths)

-sns (NTFS) alternative

-sns (Store NTFS alternate Streams) switch

Syntax
-sns[-]

Switch	Description
-sns	Enable "Store NTFS alternate streams" mode. It's default option, if you extract archive.
-sns-	Disable "Store NTFS alternate streams" mode. It's default option, if you create archive or call "list" command.

If -sns mode is enabled, 7-Zip processes NTFS Alternate Data Streams for files and folders.
Current version of 7-Zip can store NTFS alternate streams only to WIM archives.
Note: 7-Zip can't include alternate streams to archives on 32-bit Windows XP and older systems.

Examples
7z a a.wim -sns *.txt
stores txt files including alternate data streams.
7z x a.wim
unpacks a.wim including alternate data streams.
7z x a.wim -sns-
unpacks a.wim without alternate data streams.
7z l a.wim -sns
lists files in a.wim including alternate data streams.

Commands that can be used with this switch
a (Add), d (Delete), e (Extract), h (Hash), l (List), t (Test), u (Update), x (Extract with full paths)

-spf (Path File)

-spf (Use fully qualified file paths) switch

Switch	Description
-spf	Use absolute paths including drive letter.
-spf2	Use full paths without drive letter.

Enables the mode that allows to use fully qualified file paths in archives. If -spf switch is not specified, 7-Zip reduces file paths to relative paths when it adds files to archive, and 7-Zip converts paths to relative paths when you extract archive. If -spf switch is specified, 7-Zip doesn't try to process or convert paths.
Fully qualified file paths begin with one of the following:
• A UNC name, which starts with two backslash characters, for example, "\\Server1\".
• A disk designator with a backslash, for example "C:\".
• A single backslash, for example, "\Folder".
If -spf switch is specified, but the path is not fully qualified, 7-Zip will use specified path, it will not convert the path to fully qualified path.
Please be careful, if you use -spf switch with "extract" command. Check that file names in archive are correct. Note that with -spf switch 7-Zip can try to rewrite any file with path specified in archive.

Syntax
-spf

Examples
7z a a.7z -spf c:\Files\test.txt d:\test.txt
stores both txt files with full paths.
7z x a.7z -spf
extracts files from a.7z archive with exact file paths specified in archive.

Commands that can be used with this switch
a (Add), d (Delete), e (Extract), u (Update), x (Extract with full paths)

-ssw (Compress Open Writing)

-ssw (Compress files open for writing) switch
Compresses files open for writing by another applications. If this switch is not set, 7-zip doesn't include such files to archive.

Syntax
-ssw

Example
7z a archive.7z -ssw *.txt
compresses all *.txt files in current folder including files open for writing by another applications.

Commands that can be used with this switch
a (Add), h (Hash), u (Update)

-stl (Timestamp)

-stl (Set archive timestamp from the most recently modified file) switch

Syntax
-stl
If -stl switch is specified, 7-Zip sets timestamp for archive file as timestamp from the most recently modified file in that archive.

Examples
7z u a.7z -stl *.txt

Commands that can be used with this switch
a (Add), d (Delete), rn (Rename), u (Update)

-t (Type) archive

-t (set Type of archive) switch
Specifies the type of archive.

Syntax
-t{archive_type}[:s{Size}][:r][:e][:a]

{archive_type}
Specifies the type of archive. It can be: *, #, 7z, xz, split, zip, gzip, bzip2, tar, ...
*:r
Default mode. 7-Zip opens archive and subfile, if it's supported by format.
*
Opens only one top level archive.
*:s{Size}[b | k | m | g]

Sets upper limit for start of archive position. Default scan size is 8 MBytes "*:s8m". Example: "*:s0" means that it will open only file that has no any stub before archive.

#

Opens file in Parser mode, and ignores full archives.

#:a

Same as *, but it opens files with unknown extensions that contain archives in Parser Mode.

#:e

Opens file in Parser mode and checks all byte positions as start of archive. If -t{archive_type} switch is not specified, 7-Zip uses extension of archive filename to detect the type of archive. If you create new archive, -t{archive_type} switch is not specified and there is no extension of archive, 7-Zip will create .7z archive.

If -t{archive_type} switch is not specified and archive name contains incorrect extension, the program will show the warning.

It's possible to use the combined type (for example, mbr.vhd) for "Extract" and "List" commands for some archives.

When you extract archive of some types that contains another archive without compression (for example, MBR in VHD), 7-Zip can open both levels in one step. If you want to open/extract just top level archive, use -t* switch.

Note: xz, gzip and bzip2 formats support only one file per archive. If you want to compress more than one file to these formats, create a tar archive at first, and then compress it with your selected format.

Example

7z a -tzip archive.zip *.txt

adds all *.txt files from current directory to zip archive archive.zip.

7z t -t7z.split archive.7z.001

tests all files in archive.7z.001. It also checks that archive is multivolume .7z archive.

7z x -t# sfxarchive.exe

extracts sfxarchive.exe in parser mode.

7z x -tiso archive.iso

extracts files from archive.iso open as ISO archive.

7z x -tudf archive.iso

extracts files from archive.iso open as UDF archive.

Commands that can be used with this switch

a (Add), d (Delete), e (Extract), l (List), t (Test), u (Update), x (Extract with full paths)

-u (Update)

-u (Update options) switch

Specifies how to update files in an archive and (or) how to create new archives.

Syntax

-u[-]<action_set>[!{new_archive_name}]

<action_set> ::= <state_action>...

<state_action> ::= <state><action>

<state> ::= p | q | r | x | y | z | w

<action> ::= 0 | 1 | 2 | 3

Parameters

dash (-)

Disables any updates in the base archive.

The term **base archive** means the archive assigned by "base_archive_name" on the command line. See Command line syntax for more details.

{new_archive_name}

Specifies the path name of the new archive to be created. All options in this switch will refer to this new archive.

If not assigned, then all options in this switch will refer to the base archive of the command.

<state>

Specifies the state of a particular file to be processed.

<state> ::= p | q | r | x | y | z | w For each unique filename there are 6 variants of state:

<state>	State condition	File on Disk	File in Archive
p	File exists in archive, but is not matched with wildcard.		Exists, but is not matche
q	File exists in archive, but doesn't exist on disk.	Doesn't exist	Exists
r	File doesn't exist in archive, but exists on disk.	Exists	Doesn't exist
x	File in archive is newer than the file on disk.	Older	Newer
y	File in archive is older than the file on disk.	Newer	Older
z	File in archive is same as the file on disk	Same	Same
w	Can not be detected what file is newer (times are the same, sizes are different)	?	?

<action>

Specifies the action for a given <state>.

<action> ::= 0 | 1 | 2 | 3 For each state you can specify one of the three variants of actions:

<action>	Description
0	Ignore file (don't create item in new archive for this file)
1	Copy file (copy from old archive to new)
2	Compress (compress file from disk to new archive)
3	Create Anti-item (item that will delete file or directory during extracting). This feature is supported only in 7z format.

Remarks

Any update command (such as a (Add), d (Delete), u (Update)) can be assigned in these terms.
The following table shows action sets for update commands.

command \ <state>	p	q	r	x	y	z	w
d (Delete)	1	0	0	0	0	0	0
a (Add)	1	1	2	2	2	2	2
u (Update)	1	1	2	1	2	1	2
Freshen	1	1	0	1	2	1	2
Synchronize	1	0	2	1	2	1	2

If you don't specify a !{new_archive_name} option, then all options will refer to the main archive (the archive assigned on the command line after the 7z command). If you specify !{new_archive_name} option, then 7-Zip also will create a new archive with the specified name and all options will refer to that new archive.
Multiple update switches are supported. 7-Zip can create any number of new archives during one operation.
By default, the action set for each new archive is assigned as the action set of the main command. There are 3 different action sets for commands: a (Add), d (Delete), u (Update). You can overload any <state_action> pair.

Time zone notes

If you change time zone (when you move your computer to another time zone or if there are clock changes for daylight saving in your zone), you can have some problems with update commands that depend from file's modification time. It's strongly recommended to use only file system that uses Coordinated Universal Time (UTC) and archive format that also uses UTC. In that case you will have no problems with time zone changes. Also it's recommended to use only UTC formats in other cases, for example, if you send files to someone in another time zone.
Also in some cases there are no problems, if both file system and archive format use local time, for example, FAT file system and ZIP format.

- UTC file systems: NTFS
- UTC archive formats: .zip with -mtc switch, 7z, tar, gzip2, iso, wim
- Local time file systems : FAT, FAT32
- Local time archive formats : rar, zip, cab

Examples

7z u c:\1\exist.7z -u- -up0q3x2z0!c:\1\update.7z *
creates a new archive update.7z and writes to this archive all files from current directory which differ from files in exist.7z archive. exist.7z archive will not be changed.
7z u c:\1\exist.7z -up0q3x2z0!c:\1\update.7z * -ms=off
creates a new archive update.7z and writes to this archive all files from the current directory which differ from files in exist.7z archive.
Note: the updating of solid .7z archives can be slow, since it can require some recompression.

Commands that can be used with this switch

a (Add), d (Delete), rn (Rename), u (Update),

-v (Volume)

-v (Create Volumes) switch
Specifies volume sizes.

Syntax
-v{Size}[b | k | m | g]

{Size}[b | k | m | g]
Specifies volume size in Bytes, Kilobytes (1 Kilobyte = 1024 bytes), Megabytes (1 Megabyte = 1024 Kilobytes) or Gigabytes (1 Gigabyte = 1024 Megabytes). if you specify only {Size}, 7-zip will treat it as bytes.It's possible to specify several -v switches.
NOTE: Please don't use volumes (and don't copy volumes) before finishing archiving. 7-Zip can change any volume (including first volume) at the end of archiving operation.

Examples
7z a a.7z *.txt -v10k -v15k -v2m
creates multivolume a.7z archive. First volume will be 10 KB, second will be 15 KB, and all others will be 2 MB.

Commands that can be used with this switch
a (Add),

-w (Working Directory)

-w (set Working directory) switch
Sets the working directory for the temporary base archive. By default, 7-Zip builds a new base archive file in the same directory as the old base archive file. By specifying this switch, you can set the working directory where the temporary base archive file will be built. After the temporary base archive file is built, it is copied over the original archive; then, the temporary file is deleted.

Syntax
-w[{dir_path}]

{dir_path}
Specifies the destination directory path. It's not required that a path end with a backslash.
If <dir_path> is not assigned, then 7-Zip will use the Windows temporary directory.

Example
7z a -tzip archive.zip *.cpp -wc:\temp
adds *.cpp files to the archive.zip archive, creating a temporary archive in c:\temp folder.

Commands that can be used with this switch
a (Add), d (Delete), rn (Rename), u (Update),

-x (Exclude)

-x (Exclude filenames) switch
Specifies which filenames or wildcarded names must be excluded from the operation.
Multiple exclude switches are supported.

Syntax
-x[<recurse_type>]<file_ref>

<recurse_type> ::= r[- | 0]
<file_ref> ::= @{listfile} | !{wildcard}
See -i (Include) switch description for information about option parameters.

Examples
7z a -tzip archive.zip *.txt -x!temp.*

adds to the archive.zip all *.txt files, except temp.* files.
7z a archive.7z Folder1\ -xr!*.png
adds to the archive.7z all files from Folder1 and its subfolders, except *.png files.

Commands that can be used with this switch
a (Add), d (Delete), h (Hash), e (Extract), l (List), t (Test), rn (Rename), u (Update), x (Extract with full paths)

-ai (Include Archive)

-ai (Include archive filenames) switch
Specifies additional include archive filenames and wildcards.
Multiple include switches are supported.

Syntax
-ai[<recurse_type>]<file_ref>

<recurse_type> ::= r[- | 0]
<file_ref> ::= @{listfile} | !{wildcard}

Parameters

<recurse_type>
Specifies how wildcards and file names in this switch must be used. If this option is not given, recursion will be not used. For more details see specification of the -r (Recurse) switch.
<recurse_type> ::= r[- | 0]

<file_ref>
Specifies filenames and wildcards or list file that specify processed files.
<file_ref> ::= @{listfile} | !{wildcard}

Option	Description
{listfile}	Specifies name of list file. See List file description.
{wildcard}	Specifies wildcard or filename.

Examples
7z t -an -air!*.7z
tests *.7z archives in current directory and all it's subdirectories.

Commands that can be used with this switch
e (Extract), l (List), t (Test), x (Extract with full paths)

-ax (Exclude Archive)

-ax (Exclude archive filenames) switch
Specifies archives to be excluded from the operation.
Multiple exclude archive switches are supported.

Syntax
-ax[<recurse_type>]<file_ref>

<recurse_type> ::= r[- | 0]
<file_ref> ::= @{listfile} | !{wildcard}
See -xi (Include archive filenames) switch description for information about option parameters.

Examples
7z t -an -ai!*.7z -ax!a*.7z
tests all *.7z archives, except a*.7z archives.

Commands that can be used with this switch
e (Extract), l (List), t (Test), x (Extract with full paths)

-an (Disable Parsing)

-an (Disable parsing of archive_name) switch

Disables parsing of the archive_name field on the command line. This switch must be used with the -ai (Include archives) switch. If you use a file list for your archives, you specify it with the -ai switch, so you need to disable parsing of archive_name field from command line.

Syntax

-an

Examples

7z t -an -ai!*.7z -ax!a*.7z
tests all *.7z archives, except a*.7z archives.

Commands that can be used with this switch

e (Extract), l (List), t (Test), x (Extract with full paths)

-ao (Overwrite)

-ao (Overwrite mode) switch

Specifies the overwrite mode during extraction, to overwrite files already present on disk.

Syntax

-ao[a | s | t | u]

Switch	Description
-aoa	Overwrite All existing files without prompt.
-aos	Skip extracting of existing files.
-aou	aUto rename extracting file (for example, name.txt will be renamed to name_1.txt).
-aot	auto rename existing file (for example, name.txt will be renamed to name_1.txt).

Examples

7z x test.zip -aoa
extracts all files from test.zip archive and overwrites existing files without any prompt.

Commands that can be used with this switch

e (Extract), x (Extract with full paths)

-o (Output Directory)

-o (set Output directory) switch

Specifies a destination directory where files are to be extracted.
This switch can be used only with extraction commands.

Syntax

-o{dir_path}

{dir_path}
This is the destination directory path. It's not required to end with a backslash. If you specify ***** in {dir_path}, 7-Zip substitutes that * character to archive name.

Example

7z x archive.zip -oc:\Doc
extracts all files from the archive.zip archive to the c:\Doc directory.
7z x *.zip -o*
extracts all *.zip archives to subfolders with names of these archives.

Commands that can be used with this switch
e (Extract), x (Extract with full paths)

-y (YES)

-y (assume Yes on all queries) switch

Disables most of the normal user queries during 7-Zip execution. You can use this switch to suppress overwrite queries in the e (Extract) and x (Extract with full paths) commands.

Syntax

-y

Examples

7z x src.zip -y

extracts all files from src.zip archive. All overwrite queries will be suppressed and files on disk with same filenames as in archive will be overwritten.

Commands that can be used with this switch

e (Extract), x (Extract with full paths)