ccache Compiler cache

START HERE

`ccache gcc -c <file.c>`
Compile through the cache

`ccache -C`
Clear the whole cache

`ccache -s`
Show hit and miss statistics

`ccache -p`
Show effective config and origins

`ccache -z`
Zero the statistics

`CCACHE_DISABLE=1 make`
Bypass the cache for one run

`ccache -M <size>`
Set the cache size limit

`CMAKE_C_COMPILER_LAUNCHER`
CMake: set to ccache

CONCEPTS

ccache hashes the preprocessed source, compiler and flags. On a hit it returns the stored object file instead of compiling. Run a build twice and compare 'ccache -s' to see it working.

ccache gcc ... hash inputs
↓ lookup
Cache hit: copy object
↓ miss
Real compiler store result

Each compile is a lookup keyed by source, flags and compiler

Config precedence (highest first)

Source	Example
`Command line`	ccache max_size=1G gcc ...
`Environment`	CCACHE_MAXSIZE=1G
`Directory config`	ccache.conf up the tree
`Cache config`	$CCACHE_DIR/ccache.conf
`System config`	/etc/ccache.conf

RUNNING IT

`ccache <compiler> <args>`
Prefix a compiler command

`ln -s $(which ccache) ~/bin/gcc`
Masquerade as the compiler
The link directory must come before the real compiler in PATH. Repeat for g++, cc, c++, clang.

`make CC='ccache gcc'`
Pass the prefix to make

`CC="ccache gcc" ./configure`
Use it with autotools

`CMAKE_C_COMPILER_LAUNCHER`
Set to ccache via -D (also CXX)

`ccache -V`
Print the version

`ccache --help`
List all options

`ccache <KEY>=<value> gcc ...`
Set one config option for this call

STATISTICS

`ccache -s`
Show summary of hits and misses

`ccache -s -v`
Show more detailed statistics

`ccache -s -vv`
Show even more detail

`ccache -z`
Zero the statistics counters

STATISTICS (cont.)

`ccache -x`
Show compression statistics

`ccache --print-stats`
Machine-readable counters

`ccache --print-stats --format json`
Counters as JSON

`ccache --show-log-stats`
Counters from the stats log

CACHE MANAGEMENT

`ccache -M <size>`
Set maximum cache size (default GiB)

`ccache -M 0`
Remove the size limit

`ccache -F <num>`
Set maximum number of files

`ccache -c`
Clean up to the configured limits

`ccache -C`
Clear all cached files, keep config

`ccache -X <level>`
Recompress the cache at a Zstandard level

`ccache -X uncompressed`
Store the cache uncompressed

`ccache --evict-older-than <age>d`
Drop entries unused for N days

`ccache --evict-namespace <ns>`
Drop entries of one namespace

`ccache -d <dir> -s`
Operate on another cache directory

CONFIGURATION

`ccache -p`
Show effective config and its origin

`ccache -k <key>`
Print one config value

`ccache -o <key>=<value>`
Write a config option to the file

`ccache --config-path <file>`
Use another config file

`CCACHE_DIR=<dir>`
Cache location

`CCACHE_MAXSIZE=<size>`
Cache size limit (default 5GiB)

CONFIGURATION (cont.)

`CCACHE_DISABLE=1`
Bypass ccache entirely
Any set value counts as true, even an empty one. Use CCACHE_NODISABLE or unset the variable to re-enable.

`CCACHE_CONFIGPATH=<file>`
Only read this config file

`CCACHE_COMPRESS=1`
Compress cached files (on by default)

`CCACHE_COMPRESSLEVEL=<n>`
Zstandard level for new entries

`CCACHE_NAMESPACE=<name>`
Partition the cache logically

`CCACHE_UMASK=<octal>`
Permissions for shared caches

`CCACHE_STATS=0`
Stop counting statistics

HIT RATE AND CORRECTNESS

`CCACHE_BASEDIR=<dir>`
Rewrite absolute paths under dir
Needed when the same source is built from different checkout paths.

`CCACHE_SLOPPINESS=<list>`
Relax checks to get more hits

`CCACHE_SLOPPINESS=time_macros`
Ignore __DATE__ and __TIME__

`CCACHE_SLOPPINESS=pch_defines`
Allow precompiled headers

`CCACHE_COMPILERCHECK=content`
Hash compiler binary contents

`CCACHE_COMPILERCHECK=<command>`
Hash the output of a command

`CCACHE_NODIRECT=1`
Disable direct mode

`CCACHE_DEPEND=1`
Use depend mode (compiler writes deps)

`CCACHE_IGNOREOPTIONS=<opts>`
Ignore options when hashing

`CCACHE_EXTRAFILES=<files>`
Hash extra files into the key

`CCACHE_RECACHE=1`
Ignore existing results and overwrite

`CCACHE_READONLY=1`
Read from the cache, do not add

`CCACHE_COMMENTS=1`
Keep comments when hashing

HIT RATE AND CORRECTNESS (cont.)

`CCACHE_PATH=<dirs>`
Where to find the real compiler

DEBUGGING AND SHARING

`CCACHE_LOGFILE=<file>`
Log every decision to a file

`CCACHE_DEBUG=1`
Write per-object debug files

`CCACHE_DEBUGDIR=<dir>`
Directory for the debug files

`CCACHE_STATSLOG=<file>`
Collect per-build statistics

`ccache --inspect <file>`
Show a result or manifest file

`ccache --extract-result <file>`
Unpack stored files from a result

`ccache --hash-file <file>`
Print a BLAKE3 hash of a file

`CCACHE_REMOTE_STORAGE=<url>`
Add a remote cache backend

`CCACHE_REMOTE_ONLY=1`
Use only the remote storage

`CCACHE_RESHARE=1`
Push local hits to remote storage

`CCACHE_FILECLONE=1`
Use reflinks instead of copies

RECIPES

Trial: measure the speedup

```
ccache -z
make clean && make -j8
make clean && make -j8
ccache -s   # second build
should be mostly hits
```

Transparent via symlinks

```
mkdir -p ~/ccbin
for c in gcc g++ cc c++; do ln -
sf $(which ccache) ~/ccbin/$c;
done
export PATH=$HOME/ccbin:$PATH
gcc --version   # now routed
through ccache
```

CMake project with ccache

```
L=CMAKE_C_COMPILER_LAUNCHER=
ccache
X=CMAKE_CXX_COMPILER_LAUNCHER=
ccache
cmake -B build -G Ninja -D$L -
D$X
cmake --build build && ccache -s
```

Why is it missing?

```
export CCACHE_LOGFILE=/tmp/
ccache.log
make clean && make
grep -iE 'miss|result' /tmp/
ccache.log | head
ccache -p | grep -E 'sloppiness|
base_dir'
```