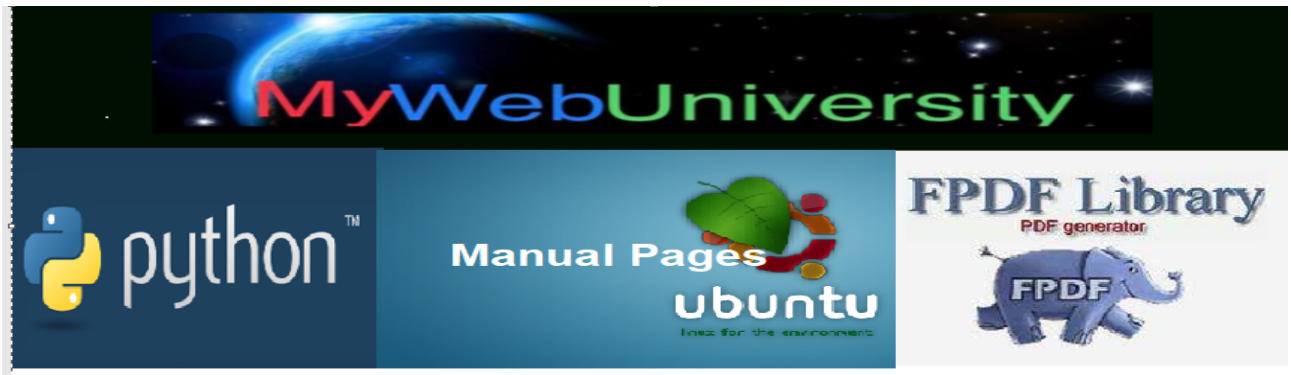

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'git-cat-file.1'

$ man git-cat-file.1

GIT-CAT-FILE(1) Git Manual GIT-CAT-FILE(1)

NAME

git-cat-file - Provide contents or details of repository objects

SYNOPSIS

git cat-file <type> <object>

git cat-file (-e | -p | -t | -s) <object>

git cat-file (--textconv | --filters)

[<rev>:<path|tree-ish> | --path=<path|tree-ish> <rev>]

git cat-file (--batch | --batch-check | --batch-command) [--batch-all-objects]

[--buffer] [--follow-symlinks] [--unordered]

[--textconv | --filters] [-Z]

DESCRIPTION

Output the contents or other properties such as size, type or delta information of one or more objects.

This command can operate in two modes, depending on whether an option from the --batch family is specified.

In non-batch mode, the command provides information on an object named on the command line.

In batch mode, arguments are read from standard input.

OPTIONS

<object>

The name of the object to show. For a more complete list of ways to spell object names, see the "SPECIFYING REVISIONS" section in gitrevisions(7).

-t

Instead of the content, show the object type identified by <object>.

-s

Instead of the content, show the object size identified by <object>. If used with --use-mailmap option, will show the size of updated object after replacing idents using the mailmap mechanism.

-e

Exit with zero status if <object> exists and is a valid object. If <object> is of an invalid format, exit with non-zero status and emit an error on stderr.

-p

Pretty-print the contents of <object> based on its type.

<type>

Typically this matches the real type of <object> but asking for a type that can trivially be dereferenced from the given <object> is also permitted. An example is to ask for a "tree" with <object> being a commit object that contains it, or to ask for a "blob" with <object> being a tag object that points at it.

--mailmap, --no-mailmap, --use-mailmap, --no-use-mailmap

Use mailmap file to map author, committer and tagger names and email addresses to canonical real names and email addresses. See git-shortlog(1).

--textconv

Show the content as transformed by a textconv filter. In this case, <object> has to be of the form <tree-ish>:<path>, or :<path> in order to apply the filter to the content recorded in the index at <path>.

--filters

Show the content as converted by the filters configured in the current working tree for the given <path> (i.e. smudge filters, end-of-line conversion, etc). In this case, <object> has to be of the form <tree-ish>:<path>, or :<path>.

--filter=<filter-spec>, --no-filter

Omit objects from the list of printed objects. This can only be used in combination with one of the batched modes. Excluded objects that have been explicitly requested via any of the batch modes that read objects via standard input (--batch, --batch-check) will be reported as "filtered". Excluded objects in --batch-all-objects mode will not be printed at all. The <filter-spec> may be one of the following:

The form --filter=blob:none omits all blobs.

The form --filter=blob:limit=<n>[kmg] omits blobs of size at least n bytes or units. n may be zero. The suffixes k, m, and g can be used to name units in KiB, MiB, or GiB. For example, blob:limit=1k is the same as blob:limit=1024.

The form --filter=object:type=(tag|commit|tree|blob) omits all objects which are not of the requested type.

--path=<path>

For use with --textconv or --filters, to allow specifying an object name and a path separately, e.g. when it is difficult to figure out the revision from which the blob came.

--batch, --batch=<format>

Print object information and contents for each object provided on stdin. May not be combined with any other options or arguments except --textconv, --filters, or --use-mailmap.

- When used with --textconv or --filters, the input lines must specify the path, separated by whitespace. See the section BATCH OUTPUT below for details.
- When used with --use-mailmap, for commit and tag objects, the contents part of the output shows the identities replaced using the mailmap mechanism, while the information part of the output shows the size of the object as if it actually recorded the replacement identities.

--batch-check, --batch-check=<format>

Print object information for each object provided on stdin. May not be combined with any other options or arguments except --textconv, --filters or --use-mailmap.

- When used with --textconv or --filters, the input lines must specify the path, separated by whitespace. See the section BATCH OUTPUT below for details.
- When used with --use-mailmap, for commit and tag objects, the printed object information shows the size of the object as if the identities recorded in it were replaced by the mailmap mechanism.

--batch-command, --batch-command=<format>

Enter a command mode that reads commands and arguments from stdin. May only be combined with --buffer, --textconv, --use-mailmap or --filters.

- When used with --textconv or --filters, the input lines must specify the path, separated by whitespace. See the section BATCH OUTPUT below for details.
- When used with --use-mailmap, for commit and tag objects, the contents command shows

the identities replaced using the mailmap mechanism, while the info command shows the size of the object as if it actually recorded the replacement identities.

--batch-command recognizes the following commands:

contents <object>

Print object contents for object reference <object>. This corresponds to the output of --batch.

info <object>

Print object info for object reference <object>. This corresponds to the output of --batch-check.

flush

Used with --buffer to execute all preceding commands that were issued since the beginning or since the last flush was issued. When --buffer is used, no output will come until a flush is issued. When --buffer is not used, commands are flushed each time without issuing flush.

--batch-all-objects

Instead of reading a list of objects on stdin, perform the requested batch operation on all objects in the repository and any alternate object stores (not just reachable objects). Requires --batch or --batch-check be specified. By default, the objects are visited in order sorted by their hashes; see also --unordered below. Objects are presented as-is, without respecting the "replace" mechanism of git-replace(1).

--buffer

Normally batch output is flushed after each object is output, so that a process can interactively read and write from cat-file. With this option, the output uses normal stdio buffering; this is much more efficient when invoking --batch-check or --batch-command on a large number of objects.

--unordered

When --batch-all-objects is in use, visit objects in an order which may be more efficient for accessing the object contents than hash order. The exact details of the order are unspecified, but if you do not require a specific order, this should generally result in faster output, especially with --batch. Note that cat-file will still show each object only once, even if it is stored multiple times in the repository.

--follow-symlinks

With --batch or --batch-check, follow symlinks inside the repository when requesting

objects with extended SHA-1 expressions of the form tree-ish:path-in-tree. Instead of providing output about the link itself, provide output about the linked-to object. If a symlink points outside the tree-ish (e.g. a link to /foo or a root-level link to ../foo), the portion of the link which is outside the tree will be printed.

This option does not (currently) work correctly when an object in the index is specified (e.g. :link instead of HEAD:link) rather than one in the tree.

This option cannot (currently) be used unless --batch or --batch-check is used.

For example, consider a git repository containing:

f: a file containing "hello\n"

link: a symlink to f

dir/link: a symlink to ../f

plink: a symlink to ../f

alink: a symlink to /etc/passwd

For a regular file f, echo HEAD:f | git cat-file --batch would print

ce013625030ba8dba906f756967f9e9ca394464a blob 6

And echo HEAD:link | git cat-file --batch --follow-symlinks would print the same thing, as would HEAD:dir/link, as they both point at HEAD:f.

Without --follow-symlinks, these would print data about the symlink itself. In the case of HEAD:link, you would see

4d1ae35ba2c8ec712fa2a379db44ad639ca277bd blob 1

Both plink and alink point outside the tree, so they would respectively print:

symlink 4

../f

symlink 11

/etc/passwd

-Z

Only meaningful with --batch, --batch-check, or --batch-command; input and output is NUL-delimited instead of newline-delimited.

-z

Only meaningful with --batch, --batch-check, or --batch-command; input is NUL-delimited instead of newline-delimited. This option is deprecated in favor of -Z as the output can otherwise be ambiguous.

If -t is specified, one of the <type>.

If -s is specified, the size of the <object> in bytes.

If -e is specified, no output, unless the <object> is malformed.

If -p is specified, the contents of <object> are pretty-printed.

If <type> is specified, the raw (though uncompressed) contents of the <object> will be returned.

BATCH OUTPUT

If --batch or --batch-check is given, cat-file will read objects from stdin, one per line, and print information about them in the same order as they have been read. By default, the whole line is considered as an object, as if it were fed to git-rev-parse(1).

When --batch-command is given, cat-file will read commands from stdin, one per line, and print information based on the command given. With --batch-command, the info command followed by an object will print information about the object the same way --batch-check would, and the contents command followed by an object prints contents in the same way --batch would.

You can specify the information shown for each object by using a custom <format>. The <format> is copied literally to stdout for each object, with placeholders of the form %(atom) expanded, followed by a newline. The available atoms are:

objectname

The full hex representation of the object name.

objecttype

The type of the object (the same as cat-file -t reports).

objectmode

If the specified object has mode information (such as a tree or index entry), the mode expressed as an octal integer. Otherwise, empty string.

objectsize

The size, in bytes, of the object (the same as cat-file -s reports).

objectsize:disk

The size, in bytes, that the object takes up on disk. See the note about on-disk sizes in the CAVEATS section below.

deltabase

If the object is stored as a delta on-disk, this expands to the full hex representation of the delta base object name. Otherwise, expands to the null OID (all zeroes). See

CAVEATS below.

rest

If this atom is used in the output string, input lines are split at the first whitespace boundary. All characters before that whitespace are considered to be the object name; characters after that first run of whitespace (i.e., the "rest" of the line) are output in place of the %(rest) atom.

If no format is specified, the default format is %(objectname) %(objecttype) %(objectsize).

If --batch is specified, or if --batch-command is used with the contents command, the object information is followed by the object contents (consisting of %(objectsize) bytes), followed by a newline.

For example, --batch without a custom format would produce:

<oid> SP <type> SP <size> LF

<contents> LF

Whereas --batch-check='%(objectname) %(objecttype)' would produce:

<oid> SP <type> LF

If a name is specified on stdin that cannot be resolved to an object in the repository, then cat-file will ignore any custom format and print:

<object> SP missing LF

If a name is specified on stdin that is filtered out via --filter=, then cat-file will ignore any custom format and print:

<object> SP excluded LF

If a name is specified that might refer to more than one object (an ambiguous short sha), then cat-file will ignore any custom format and print:

<object> SP ambiguous LF

If a name is specified that refers to a submodule entry in a tree and the target object does not exist in the repository, then cat-file will ignore any custom format and print (with the object ID of the submodule):

<oid> SP submodule LF

If --follow-symlinks is used, and a symlink in the repository points outside the repository, then cat-file will ignore any custom format and print:

symlink SP <size> LF

<symlink> LF

The symlink will either be absolute (beginning with a /), or relative to the tree root. For

instance, if dir/link points to ../../foo, then <symlink> will be ../foo. <size> is the size of the symlink in bytes.

If --follow-symlinks is used, the following error messages will be displayed:

<object> SP missing LF

is printed when the initial symlink requested does not exist.

dangling SP <size> LF

<object> LF

is printed when the initial symlink exists, but something that it (transitive-of) points to does not.

loop SP <size> LF

<object> LF

is printed for symlink loops (or any symlinks that require more than 40 link resolutions to resolve).

notdir SP <size> LF

<object> LF

is printed when, during symlink resolution, a file is used as a directory name.

Alternatively, when -Z is passed, the line feeds in any of the above examples are replaced with NUL terminators. This ensures that output will be parsable if the output itself would contain a linefeed and is thus recommended for scripting purposes.

CAVEATS

Note that the sizes of objects on disk are reported accurately, but care should be taken in drawing conclusions about which refs or objects are responsible for disk usage. The size of a packed non-delta object may be much larger than the size of objects which delta against it, but the choice of which object is the base and which is the delta is arbitrary and is subject to change during a repack.

Note also that multiple copies of an object may be present in the object database; in this case, it is undefined which copy?s size or delta base will be reported.

GIT

Part of the git(1) suite