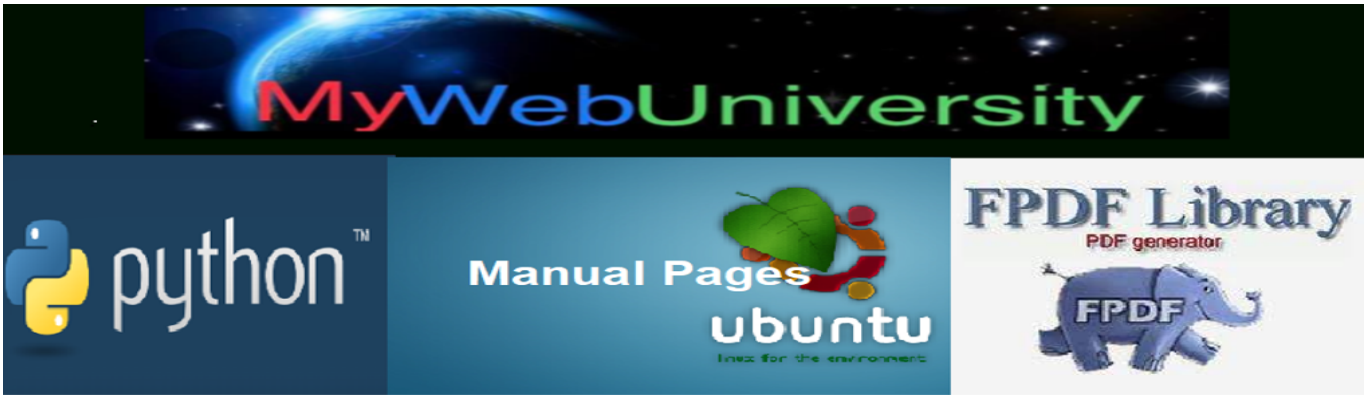

Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!

Rocky Enterprise Linux 9.2 Manual Pages on command 'git-hash-object.1'

$ man git-hash-object.1

GIT-HASH-OBJECT(1) Git Manual GIT-HASH-OBJECT(1)

NAME

git-hash-object - Compute object ID and optionally creates a blob from a file

SYNOPSIS

git hash-object [-t <type>] [-w] [--path=<file>|--no-filters] [--stdin [--literally]] [--] <file>...

git hash-object [-t <type>] [-w] --stdin-paths [--no-filters]

DESCRIPTION

Computes the object ID value for an object with specified type with the contents of the named file (which can be outside of the work tree), and optionally writes the resulting object into the object database. Reports its object ID to its standard output. When <type> is not specified, it defaults to "blob".

OPTIONS

-t <type>

Specify the type (default: "blob").

-w

Actually write the object into the object database.

--stdin

Read the object from standard input instead of from a file.

--stdin-paths

Read file names from the standard input, one per line, instead of from the command-line.

--path

Hash object as it were located at the given path. The location of file does not

directly influence on the hash value, but path is used to determine what Git filters should be applied to the object before it can be placed to the object database, and, as result of applying filters, the actual blob put into the object database may differ from the given file. This option is mainly useful for hashing temporary files located outside of the working directory or files read from stdin.

--no-filters

Hash the contents as is, ignoring any input filter that would have been chosen by the attributes mechanism, including the end-of-line conversion. If the file is read from standard input then this is always implied, unless the --path option is given.

--literally

Allow --stdin to hash any garbage into a loose object which might not otherwise pass standard object parsing or git-fsck checks. Useful for stress-testing Git itself or reproducing characteristics of corrupt or bogus objects encountered in the wild.

GIT

Part of the git(1) suite