NAME

git-archive - Create an archive of files from a named tree

SYNOPSIS

git archive [--format=<fmt>] [--list] [--prefix=<prefix>/] [<extra>]
[-o <file> | --output=<file>] [--worktree-attributes]
[--remote=<repo> [--exec=<git-upload-archive>]] <tree-ish>
[<path>...]

DESCRIPTION

Creates an archive of the specified format containing the tree structure for the named tree, and writes it out to the standard output. If <prefix> is specified it is prepended to the filenames in the archive.

git archive behaves differently when given a tree ID versus when given a commit ID or tag ID. In the first case the current time is used as the modification time of each file in the archive. In the latter case the commit time as recorded in the referenced commit object is used instead. Additionally the commit ID is stored in a global extended pax header if the tar format is used; it can be extracted using *git get-tar-commit-id*. In ZIP files it is stored as a file comment.

OPTIONS

--format=<fmt>
: Format of the resulting archive: *tar* or *zip*. If this option is not given, and the output file is specified, the format is inferred from the filename if possible (e.g. writing to foo.zip makes the output to be in the zip format). Otherwise the output format is tar.

-l, --list
: Show all available formats.

-v, --verbose
: Report progress to stderr.

--prefix=<prefix>/
: Prepend <prefix>/ to each filename in the archive.

-o <file>, --output=<file>
: Write the archive to <file> instead of stdout.

--worktree-attributes
: Look for attributes in .gitattributes files in the working tree as well (see the section called "ATTRIBUTES").

<extra>
: This can be any options that the archiver backend understands. See next section.

--remote=<repo>
: Instead of making a tar archive from the local repository, retrieve a tar archive from a remote repository. Note that the remote repository may place restrictions on which sha1 expressions may be allowed in <tree-ish>. See **git-upload-archive(1)** for details.

--exec=<git-upload-archive>
: Used with --remote to specify the path to the *git-upload-archive* on the remote side.

<tree-ish>
: The tree or commit to produce an archive for.

<path>
: Without an optional path parameter, all files and subdirectories of the current working directory are included in the archive. If one or more paths are specified, only these are included.

BACKEND EXTRA OPTIONS

zip

-0
: Store the files instead of deflating them.

-9
: Highest and slowest compression level. You can specify any number from 1 to 9 to adjust compression speed and ratio.

CONFIGURATION

tar.umask
: This variable can be used to restrict the permission bits of tar archive entries. The default is 0002, which turns off the world write bit. The special value user indicates that the archiving user's umask will be used instead. See umask(2) for details. If --remote is used then only the configuration of the remote repository takes effect.

tar.<format>.command
: This variable specifies a shell command through which the tar output generated by git archive should be piped. The command is executed using the shell with the generated tar file on its standard input, and should produce the final output on its standard output. Any compression-level options will be passed to the command (e.g., -9). An output file with the same extension as <format> will be use this format if no other format is given.

  The tar.gz and tgz formats are defined automatically and default to gzip -cn. You may override them with custom commands.

tar.<format>.remote
: If true, enable <format> for use by remote clients via **git-upload-archive(1)**. Defaults to false for user-defined formats, but true for the tar.gz and tgz formats.

ATTRIBUTES

export-ignore
: Files and directories with the attribute export-ignore won't be added to archive files. See **gitattributes(5)** for details.

export-subst
: If the attribute export-subst is set for a file then Git will expand several placeholders when adding this file to an archive. See **gitattributes(5)** for details.

Note that attributes are by default taken from the .gitattributes files in the tree that is being archived. If you want to tweak the way the output is generated after the fact (e.g. you committed without adding an appropriate export-ignore in its .gitattributes), adjust the checked out .gitattributes file as necessary and use --worktree-attributes option. Alternatively you can keep necessary attributes that should apply while archiving any tree in your $GIT_DIR/info/attributes file.

EXAMPLES

git archive --format=tar --prefix=junk/ HEAD | (cd /var/tmp/ && tar xf -)
: Create a tar archive that contains the contents of the latest commit on the current branch, and extract it in the /var/tmp/junk directory.

git archive --format=tar --prefix=git-1.4.0/ v1.4.0 | gzip >git-1.4.0.tar.gz
: Create a compressed tarball for v1.4.0 release.

git archive --format=tar.gz --prefix=git-1.4.0/ v1.4.0 >git-1.4.0.tar.gz
: Same as above, but using the builtin tar.gz handling.

git archive --prefix=git-1.4.0/ -o git-1.4.0.tar.gz v1.4.0
: Same as above, but the format is inferred from the output file.

git archive --format=tar --prefix=git-1.4.0/ v1.4.0^{tree} | gzip >git-1.4.0.tar.gz
: Create a compressed tarball for v1.4.0 release, but without a global extended pax header.

git archive --format=zip --prefix=git-docs/ HEAD:Documentation/ > git-1.4.0-docs.zip
: Put everything in the current head's Documentation/ directory into *git-1.4.0-docs.zip*, with the prefix *git-docs/*.

git archive -o latest.zip HEAD
: Create a Zip archive that contains the contents of the latest commit on the current branch. Note that the output format is inferred by the extension of the output file.

git config tar.tar.xz.command xz -c
: Configure a tar.xz format for making LZMA-compressed tarfiles. You can use it specifying --format=tar.xz, or by creating an output file like -o foo.tar.xz.

SEE ALSO

gitattributes(5)

GIT

Part of the **git(1)** suite