NAME

hardlink − link multiple copies of a file

SYNOPSIS

hardlink [options] [*directory*|*file*]...

DESCRIPTION

hardlink is a tool that replaces copies of a file with either hardlinks or copy−on−write clones, thus saving space.

hardlink first creates a binary tree of file sizes and then compares the content of files that have the same size. There are two basic content comparison methods. The **memcmp** method directly reads data blocks from files and compares them. The other method is based on checksums (like SHA256); in this case for each data block a checksum is calculated by the Linux kernel crypto API, and this checksum is stored in userspace and used for file comparisons.

For each file also an "intro" buffer (32 bytes) is cached. This buffer is used independently from the comparison method and requested cache−size and io−size. The "intro" buffer dramatically reduces operations with data content as files are very often different from the beginning.

OPTIONS

−h, **−−help**

Display help text and exit.

−V, **−−version**

Print version and exit.

−v, **−−verbose**

Verbose output, explain to the user what is being done. If specified once, every hardlinked file is displayed. If specified twice, it also shows every comparison.

−q, **−−quiet**

Quiet mode, don’t print anything.

−n, **−−dry−run**

Do not act, just print what would happen.

−y, **−−method** *name*

Set the file content comparison method. The currently supported methods are sha256, sha1, crc32c and memcmp. The default is sha256, or memcmp if Linux Crypto API is not available. The methods based on checksums are implemented in zero−copy way, in this case file contents are not copied to the userspace and all calculation is done in kernel.

−−reflink[=*when*]

Create copy−on−write clones (aka reflinks) rather than hardlinks. The reflinked files share only on−disk data, but the file mode and owner can be different. It’s recommended to use it with **−−ignore−owner** and **−−ignore−mode** options. This option implies **−−skip−reflinks** to ignore already cloned files.

The optional argument *when* can be **never**, **always**, or **auto**. If the *when* argument is omitted, it defaults to **auto**, in this case, **hardlink** checks filesystem type and uses reflinks on BTRFS and XFS only, and fallback to hardlinks when creating reflink is impossible. The argument **always** disables filesystem type detection and fallback to hardlinks, in this case, only reflinks are allowed.

−−skip−reflinks

Ignore already cloned files. This option may be used without **−−reflink** when creating classic hardlinks.

−f, **−−respect−name**
Only try to link files with the same (base)name. It’s strongly recommended to use long options rather than **−f** which is interpreted in a different way by other **hardlink** implementations.

−p, **−−ignore−mode**
Link and compare files even if their mode is different. Results may be slightly unpredictable.

−o, **−−ignore−owner**
Link and compare files even if their owner information (user and group) differs. Results may be unpredictable.

−t, **−−ignore−time**
Link and compare files even if their time of modification is different. This is usually a good choice.

−c −−content
Consider only file content, not attributes, when determining whether two files are equal. Same as **−pot**.

−X, **−−respect−xattrs**
Only try to link files with the same extended attributes.

−m, **−−maximize**
Among equal files, keep the file with the highest link count.

−M, **−−minimize**
Among equal files, keep the file with the lowest link count.

−O, **−−keep−oldest**
Among equal files, keep the oldest file (least recent modification time). By default, the newest file is kept. If **−−maximize** or **−−minimize** is specified, the link count has a higher precedence than the time of modification.

−x, **−−exclude** *regex*
A regular expression which excludes files from being compared and linked.

−i, **−−include** *regex*
A regular expression to include files. If the option **−−exclude** has been given, this option re−includes files which would otherwise be excluded. If the option is used without **−−exclude**, only files matched by the pattern are included.

−s, **−−minimum−size** *size*
The minimum size to consider. By default this is 1, so empty files will not be linked. The *size* argument may be followed by the multiplicative suffixes KiB (=1024), MiB (=1024*1024), and so on for GiB, TiB, PiB, EiB, ZiB and YiB (the "iB" is optional, e.g., "K" has the same meaning as "KiB").

−S, **−−maximum−size** *size*
The maximum size to consider. By default this is 0 and 0 has the special meaning of unlimited. The *size* argument may be followed by the multiplicative suffixes KiB (=1024), MiB (=1024*1024), and so on for GiB, TiB, PiB, EiB, ZiB and YiB (the "iB" is optional, e.g., "K" has the same meaning as "KiB").

−b, **−−io−size** *size*
The size of the **read**(2) or **sendfile**(2) buffer used when comparing file contents. The *size* argument

may be followed by the multiplicative suffixes KiB, MiB, etc. The "iB" is optional, e.g., "K" has the same meaning as "KiB". The default is 8KiB for memcmp method and 1MiB for the other methods. The only memcmp method uses process memory for the buffer, other methods use zero−copy way and I/O operation is done in the kernel. The size may be altered on the fly to fit a number of cached content checksums.

−r, **−−cache−size** *size*

The size of the cache for content checksums. All non−memcmp methods calculate checksum for each file content block (see **−−io−size**), these checksums are cached for the next comparison. The size is important for large files or a large sets of files of the same size. The default is 10MiB.

ARGUMENTS

hardlink takes one or more directories which will be searched for files to be linked.

BUGS

The original **hardlink** implementation uses the option **−f** to force hardlinks creation between filesystem. This very rarely usable feature is no more supported by the current **hardlink**.

hardlink assumes that the trees it operates on do not change during operation. If a tree does change, the result is undefined and potentially dangerous. For example, if a regular file is replaced by a device, **hardlink** may start reading from the device. If a component of a path is replaced by a symbolic link or file permissions change, security may be compromised. Do not run **hardlink** on a changing tree or on a tree controlled by another user.

AUTHOR

There are multiple **hardlink** implementations. The very first implementation is from Jakub Jelinek for Fedora distribution, this implementation has been used in util−linux between versions v2.34 to v2.36. The current implementations is based on Debian version from Julian Andres Klode.

REPORTING BUGS

For bug reports, use the issue tracker at https://github.com/util−linux/util−linux/issues.

AVAILABILITY

The **hardlink** command is part of the util−linux package which can be downloaded from Linux Kernel Archive <https://www.kernel.org/pub/linux/utils/util−linux/>.