## NAME

wimlib-imagex-apply − Extract one image, or all images, from a WIM archive

## SYNOPSIS

**wimlib-imagex apply** *WIMFILE* [*IMAGE*] *TARGET* [*OPTION*...]

## DESCRIPTION

**wimlib-imagex apply** extracts an image, or all images, from the Windows Imaging (WIM) file *WIMFILE*.

This command is designed to extract, or "apply", one or more full WIM images. If you instead want to extract only certain files or directories contained in a WIM image, consider using **wimlib-imagex extract** or **wimlib-imagex mount** instead.

*IMAGE* specifies the WIM image to extract. It may be a 1-based index of an image in the WIM, the name of an image in the WIM, or the keyword "all" to indicate that all images are to be extracted. Use the **wimlib-imagex info** (1) command to show what images a WIM file contains. *IMAGE* may be omitted if *WIMFILE* contains only one image.

*TARGET* specifies where to extract the WIM image(s) to. If *TARGET* specifies a directory, the WIM image(s) are extracted to that directory. If *TARGET* specifies a non-existent file, a directory is created in that location and the WIM image(s) are extracted to that directory. Alternatively, on UNIX only, if *TARGET* specifies a regular file or block device, it is interpreted as an NTFS volume to which the WIM image is to be extracted.

**wimlib-imagex apply** supports applying images from stand-alone WIMs as well as split WIMs. See **SPLIT WIMS**.

## NORMAL MODE (UNIX)

This section documents how files are extracted on UNIX from the WIM image to a directory. See **WINDOWS VERSION** for the corresponding documentation for the Windows version.

On UNIX, the "normal" extraction mode is entered when *TARGET* is a directory or non-existent file. If a single WIM image is being extracted, it is extracted with the root directory of the image corresponding to the directory named by *TARGET*; or, if the keyword **all** is given, the images are extracted into subdirectories of *TARGET* that are be named after the image names, falling back to the image index for an image with no name. *TARGET* can specify a directory on any type of filesystem.

In the "normal" mode of extraction on UNIX, the following information is extracted from the WIM image(s):

- The default (unnamed) data stream of each file
- Hard links
- File and directory creation, access, and modification timestamps to the nearest 100 nanoseconds, if supported by the underlying filesystem, operating system, and C library
- Symbolic links and junction points. Drive letters will be stripped. (Note: see **--rpfix** and **--norpfix** for documentation on how absolute symbolic links and junctions are applied.)

However, in the "normal" mode of extraction on UNIX, the following information will *not* be extracted from the WIM image(s):

- Security descriptors (file permissions) except through the extensions available through the **--unix-data** option
- The alternate (named) data streams for each file
- Reparse points other than symbolic links and junction points
- Certain file attributes such as compression, encryption, and sparseness.
- Short (DOS) names for files

## NTFS MODE (UNIX)

This section documents how files are extracted directly to an NTFS volume image on UNIX. See **WINDOWS VERSION** for the corresponding documentation for the Windows version.

On UNIX, a special extraction mode is entered when *TARGET* is a regular file or block device. If this is the case, *TARGET* is interpreted as an NTFS volume and opened using libntfs-3g. If successful, the WIM image is extracted to the root of the NTFS volume in a special mode that preserves all information contained in the WIM image. *IMAGE* may not be "all" for this action.

The NTFS volume does not need to be empty, although it's expected that it be empty for the intended use cases. A new NTFS filesystem can be created using the **mkntfs** (8) command.

The NTFS extraction mode is not available if wimlib was compiled using the **--without-ntfs-3g** option.

Please note that the NTFS extraction mode is *not* entered if *TARGET* is a directory, even if an NTFS filesystem is mounted on *TARGET*. You must specify the NTFS volume itself (and it must be unmounted, and you must have permission to write to it).

In the NTFS extraction mode on UNIX, the following information will be extracted from the WIM image:

- The data streams of all files, including the un-named data stream as well as all named data streams.
- Reparse points, including symbolic links, junction points, and other reparse points.
- Hard links.
- File and directory creation, access, and modification timestamps are set to the 100-nanosecond resolution values specified in the WIM file.
- The security descriptor for each file is applied if there is one specified in the WIM.
- File attribute flags are applied.
- Short (DOS) names for files are extracted. The corresponding long name for each DOS name is made to be a Win32 name. Any additional names for the file in the same directory are made to be names in the POSIX namespace.

However, the extraction of encrypted files is not supported in this mode.

Since all (or almost all) information from the WIM image is restored in this mode, it is possible to restore an image of an actual Windows installation using **wimlib-imagex** on UNIX in addition to with **wimlib-imagex** on Windows. In the examples at the end of this manual page, there is an example of applying an image from the "install.wim" file contained in the installation media for Windows Vista, Windows 7, and Windows 8 in the "sources" directory.

But in order to actually boot Windows from an applied image, you must understand the boot process of Windows versions Vista and later. Basically, it is the following:

1. The Master Boot Record loads the Volume Boot Record (also called the Boot Sector) of the active partition, which is on an NTFS filesystem. This partition is called the "system partition".
2. The "bootmgr" program on the "system partition" is loaded (\BOOTMGR).
3. bootmgr loads the Boot Configuration Data (\Boot\BCD) from the "system partition".
4. Based on the information contained in the Boot Configuration Data, a loader for the Windows kernel is executed from the "Boot" partition, which is where Windows is installed.

So let's say you applied an image from an existing "install.wim" as in the example, or you've applied a custom Windows image that you've created using the **wimlib-imagex capture** (1) command. You've just applied the "Boot" partition, or the main Windows partition, but there is no "System" partition yet (i.e. no \BOOTMGR and no \Boot\BCD).

A "System" partition can be created created by running the "bcdboot.exe" program from within Windows or Windows PE. Alternatively, you can capture a separate WIM image containing the "System" partition. Or, the "System" partition may the same as the "Boot" partition, so the two "partitions" may be combined in one WIM image. However, as the \Boot\BCD file contains the Windows bootloader configuration, a WIM

containing it can only be used on systems where you are setting up the same bootloader configuration, including the same partition layout.

Besides setting up the files on the "System" partition, don't forget to set the bootable flag on it, and have a master boot record that loads the bootable partition (Windows' MBR does, and SYSLINUX provides an equivalent MBR).

## WINDOWS VERSION

The Windows version of **wimlib-imagex apply** acts similarly to the corresponding command of Microsoft's ImageX. For best results, the target directory should be on an NTFS volume and you should be running with Administrator privileges; however, non-NTFS filesystems and running without Administrator privileges are also supported.

On Windows, **wimlib-imagex apply** tries to extract as much data as possible. This includes:

- All data streams of all files. This includes the default file contents, as well as named data streams if supported by the filesystem.
- Reparse points, including symbolic links, junction points, and other reparse points, if supported by the underlying filesystem. (Note: see **--rpfix** and **--norpfix** for documentation on how absolute symbolic links and junctions are applied.)
- File and directory creation, access, and modification timestamps.
- Security descriptors, if supported by the filesystem and **--no-acls** is not specified. Furthermore, unless **--strict-acls** is specified, the security descriptor for individual files or directories may be omitted or only partially set if the user does not have permission to set them.
- File attributes, including hidden, sparse, compressed, encrypted, etc, when supported by the filesystem.
- DOS names (8.3) names of files; however, the failure to set them is not considered an error condition.
- Hard links, if supported by the filesystem.

Additional notes about extracting files on Windows:

- Encrypted files will be extracted as raw encrypted data if the filesystem does not support encryption.
- Compressed files and directories (with the compression attribute set) will be extracted as uncompressed if the filesystem does not support transparent compression.
- Files with names that cannot be represented on Windows will not be extracted by default; see **--include-invalid-names**.
- Files with full paths over 260 characters (MAX_PATH) are extracted by using the \\?\-prefixed path hack. But beware that such files will be inaccessible to most Windows software and may not be able to be deleted easily.

## SPLIT WIMS

You may use **wimlib-imagex apply** to apply images from a split WIM. The *WIMFILE* argument is used to specify the first part of the split WIM, and the **--refs**="*GLOB*" option is used to provide a shell-style file glob that specifies the additional parts of the split WIM. *GLOB* is expected to be a single string on the command line, so *GLOB* must be quoted so that it is protected against shell expansion. *GLOB* must expand to all parts of the split WIM, except optionally the first part which may either omitted or included in the glob (but the first part MUST be specified as *WIMFILE* as well).

Here's an example. The names for the split WIMs usually go something like:

```
mywim.swm
mywim2.swm
mywim3.swm
mywim4.swm
mywim5.swm
```

To apply the first image of this split WIM to the directory "dir", run:

wimlib-imagex apply mywim.swm 1 dir --ref="mywim*.swm"

## OPTIONS

**--check**

When reading *WIMFILE*, verify its integrity if the integrity table is present.

**--ref**="*GLOB*"

File glob of additional split WIM parts that are part of the split WIM being applied. See **SPLIT_WIMS**.

**--rpfix**, **--norpfix**

Set whether to fix targets of absolute symbolic links (reparse points in Windows terminology) or not. When enabled (**--rpfix**), extracted absolute symbolic links that are marked in the WIM image as being fixed are assumed to have absolute targets relative to the image root, and therefore have the actual root of extraction prepended to their targets. The intention is that you can apply an image containing absolute symbolic links and still have them be valid after it has been applied to any location.

The default behavior is **--rpfix** if any images in *WIMFILE* have been captured with reparse-point fixups done. Otherwise, it is **--norpfix**.

Reparse point fixups are never done in the NTFS extraction mode on UNIX.

**--verbose**

Print the path to of each file or directory within the WIM image as it is extracted.

**--hardlink**

(UNIX only) When extracting a file from the WIM that is identical to a file that has already extracted, create a hard link rather than creating a separate file. This option causes all identical files to be hard-linked, overriding the hard link groups that are specified in the WIM image(s). In the case of extracting all images from the WIM, files may be hard-linked even if they are in different WIM images. This option is not available in the NTFS extraction mode.

**--symlink**

(UNIX only) This option is similar to **--hardlink**, except symbolic links are created instead.

**--unix-data**

(UNIX only) By default, in the normal extraction mode on UNIX, **wimlib-imagex apply** will ignore both Windows-style security descriptors and UNIX-specific file owners, groups, and modes set when using **wimlib-imagex capture** with the **--unix-data** flag. By passing **--unix-data** to **wimlib-imagex apply** instead, this causes this UNIX-specific data to be restored when available. However, by default, if **wimlib-imagex** does not have permission to set the UNIX owner, group or file mode on an extracted file, a warning will be printed and it will not be considered an error condition; use **--strict-acls** to get stricter behavior.

**--no-acls**

(Windows only) Do not restore security descriptors on extracted files and directories.

**--strict-acls**

On Windows: Fail immediately if the full security descriptor of any file or directory cannot be set exactly as specified in the WIM file. The default behavior without this option is to fall back to setting a security descriptor with the SACL omitted, then only the default inherited security descriptor, if we do not have permission to set the desired one. On UNIX: with **--unix-data**, fail immediately if the UNIX owner, group, or file mode on an extracted file cannot be set for any reason.

**--include-invalid-names**

Extract files and directories with invalid names by replacing characters and appending a suffix rather than ignoring them. The meaning of this is platform-dependent.

On UNIX, filenames are case-sensitive and may contain any byte except '\0' and ´/', so on UNIX this option will only have an effect in the unlikely case that the WIM image for some reason has a filename containing one of these characters.

On Windows, filenames are case-insensitive, cannot include the characters ’/’, ´\0’, ’\’, ’:’, ’*’, ’?’, ’"’, ’<’, ’>’, or ’|’, and cannot end with a space or period. Ordinarily, files in WIM images should meet these conditions as well. However, it is not guaranteed, and in particular a WIM image captured with **wimlib-imagex** on UNIX could contain such files. By default, invalid names will be ignored, and if there are multiple names differing only in case, one will be chosen to extract arbitrarily; however, with **--include-invalid-names**, all names will be sanitized and extracted in some form.

## NOTES

**wimlib-imagex apply** calculates the SHA1 message digest of every file stream it extracts and verifies that it is the same as the SHA1 message digest provided in the WIM file. It is an error if the message digests don’t match. It’s also considered to be an error if any WIM resources cannot be found in the stream lookup table. So you can be fairly certain that the file streams are extracted correctly, even though **wimlib-imagex apply** don’t have a **/verify** option like Microsoft’s ImageX does. Please note that this is separate from the integrity table of the WIM, which provides SHA1 message digests over raw chunks of the entire WIM file and is checked separately if the **--check** option is specified.

You cannot use **wimlib-imagex apply** to apply a WIM from a pipe (such as standard input) because the WIM file format is not designed for this.

## EXAMPLES

### Applying a WIM image to a directory (both UNIX and Windows)

Extract the first image from the Windows PE image from the Windows Vista/7/8 installation media to the directory "boot":

```
wimlib-imagex apply /media/windows/sources/boot.wim 1 boot
```

Extract all images from the Windows PE image from the Windows Vista/7/8 installation media to the directory "boot", and hard link all identical files:

```
wimlib-imagex apply /media/windows8/sources/boot.wim all boot --hardlink
```

### Applying a WIM image directly to a NTFS volume image (UNIX only)

Apply a WIM image to an NTFS filesystem image:

```
wimlib-imagex apply mywim.wim 1 fsimage.ntfs
```

Create a new NTFS filesystem on the partition /dev/sda2 and apply the first image in the Windows Vista/7/8 installation WIM to it. (Obviously, only do this if you want to erase everything on that partition.)

```
mkntfs /dev/sda2 && wimlib-imagex apply /media/windows/sources/install.wim 1 /dev/sda2
```

## SEE ALSO

**wimlib-imagex**(1) **wimlib-imagex-extract**(1) **wimlib-imagex-info**(1)