NAME

wimcapture, wimappend – Capture or append a WIM image

SYNOPSIS

wimcapture *SOURCE WIMFILE* [*IMAGE_NAME* [*IMAGE_DESC*]] [*OPTION*...]
wimappend *SOURCE WIMFILE* [*IMAGE_NAME* [*IMAGE_DESC*]] [*OPTION*...]

DESCRIPTION

The **wimcapture** (equivalently: **wimlib-imagex capture**) and **wimappend** (equivalently: **wimlib-imagex append**) commands create ("capture") a new Windows Imaging (WIM) image. **wimcapture** creates a new WIM archive *WIMFILE* to contain the new image, while **wimappend** adds the image to the existing WIM archive *WIMFILE* (or with **--create**, creating it if needed).

SOURCE specifies the location of the files from which to create the WIM image. If *SOURCE* is a directory or a symbolic link pointing to a directory, then the image is captured from that directory as per **DIRECTORY CAPTURE (UNIX)** or **DIRECTORY CAPTURE (WINDOWS)**. Alternatively, if **--source-list** is specified, then *SOURCE* is interpreted as a file containing a list of files and directories to include in the image. Still alternatively, if *SOURCE* is a UNIX block device, then an image is captured from the NTFS volume on it as per **NTFS VOLUME CAPTURE (UNIX)**.

IMAGE_NAME and *IMAGE_DESC* specify the name and description to give the new image. If *IMAGE_NAME* is unspecified, it defaults to the filename component of *SOURCE*, appending a unique suffix if needed. Otherwise, *IMAGE_NAME* must be either a name not already used by an image in *WIMFILE*, or the empty string to create an unnamed image. If *IMAGE_DESC* is unspecified then the new image is given no description.

If *WIMFILE* is specified as "-", then the **--pipable** option is assumed and a pipable WIM is written to standard output (this is a wimlib extension).

DIRECTORY CAPTURE (UNIX)

On UNIX-like systems, if *SOURCE* specifies a directory or a symbolic link to a directory, then the WIM image will be captured from that directory. The directory can be on any type of filesystem, and mountpoints are followed. In this mode, the following types of information are captured:

- Directories and regular files, and the contents of regular files
- Hard links
- Symbolic links (translated losslessly to Windows reparse points)
- Last modification times (mtime) and last access times (atime) with 100 nanosecond granularity
- Files that appear to be sparse will be flagged as such, but their full data will still be stored, subject to the usual compression.
- With **--unix-data**: standard UNIX file permissions (owner, group, and mode)
- With **--unix-data**: device nodes, named pipes, and sockets
- With **--unix-data**: extended attributes (Linux only)

There is no support for storing last status change times (ctimes), or hard link information for symlinks (each symlink will be stored as a separate file). Also, filenames and symlink targets which are not valid UTF-8 with the addition of surrogate codepoints are unsupported. Note that if you have a filesystem containing filenames in another multibyte encoding, such as ISO-8859-1, and you wish to archive it with wimlib, you may be able to mount it with an option which causes its filenames to be presented as UTF-8.

NTFS VOLUME CAPTURE (UNIX)

On UNIX-like systems, *SOURCE* may also be specified as a block device (e.g. /dev/sda3) containing an unmounted NTFS volume. In this mode, **wimcapture** uses libntfs-3g to capture a WIM image from root directory of the NTFS volume. In this mode, as much data and metadata as possible is captured, including NTFS-specific and Windows-specific metadata:

- All data streams of all unencrypted files, including the unnamed data stream as well as all named data streams.
- Reparse points. See **REPARSE POINTS AND SYMLINKS** for details.
- File and directory creation, access, and modification timestamps, using the native NTFS resolution of 100 nanoseconds.
- Windows security descriptors, including all components (owner, group, DACL, and SACL).
- DOS/Windows file attribute flags.
- All names of all files, including names in the Win32 namespace, DOS namespace, Win32+DOS namespace, and POSIX namespace. This includes hard links.
- Object IDs.

However, the main limitations of this mode are:

- Encrypted files are excluded.
- Extended attributes (EAs) are not stored yet.
- Sparse files will be flagged as such, but their full data will still be stored, subject to the usual compression.
- Some types of reparse points are transparently dereferenced by Windows but not by NTFS-3G. See **REPARSE POINTS AND SYMLINKS**.

Note that this mode uses libntfs-3g directly, without going through the **ntfs-3g**(8) driver. Hence, there is no special support for capturing a WIM image from a directory on which an NTFS filesystem has been mounted using **ntfs-3g**(8); you have to unmount it first. There is also no support for capturing a subdirectory of the NTFS volume; you can only capture the full volume.

DIRECTORY CAPTURE (WINDOWS)

On Windows, **wimcapture** and **wimappend** natively support Windows-specific and NTFS-specific data. They therefore act similarly to the corresponding commands of Microsoft's ImageX or DISM. For best results, the directory being captured should be on an NTFS volume and the program should be run with Administrator privileges; however, non-NTFS filesystems and running without Administrator privileges are also supported, subject to limitations.

On Windows, **wimcapture** and **wimappend** try to capture as much data and metadata as possible, including:

- All data streams of all files.
- Reparse points, if supported by the source filesystem. See **REPARSE POINTS AND SYMLINKS** for details.
- File and directory creation, access, and modification timestamps. These are stored with Windows' native timestamp resolution of 100 nanoseconds.
- Security descriptors, if supported by the source filesystem and **--no-acls** is not specified. Note: when not running as an Administrator, security descriptors may be only partially captured (see **--strict-acls**).
- File attributes, including hidden, sparse, compressed, encrypted, etc. Encrypted files will be stored in encrypted form rather than in plain text. Transparently compressed files will be read as uncompressed and stored subject to the WIM's own compression. There is no special handling for storing sparse files, but they are likely to compress to a small size.
- DOS names (8.3) names of files; however, the failure to read them is not considered an error condition.
- Hard links, if supported by the source filesystem.
- Object IDs, if supported by the source filesystem.

- Extended attributes (EAs), if supported by the source filesystem.

REPARSE POINTS AND SYMLINKS

A "symbolic link" (or "symlink") is a special file which "points to" some other file or directory. On Windows, a "reparse point" is a generalization of a symlink which allows access to a file or directory to be redirected in a more complex way. Windows uses reparse points to implement symlinks and sometimes uses them for various other features as well. Normally, applications can choose whether they want to "dereference" reparse points and symlinks or not.

The default behavior of **wimcapture** is that reparse points and symlinks are *not* dereferenced, meaning that the reparse points or symlinks themselves are stored in the archive rather than the files or data they point to. There is a **--dereference** option, but it is currently only supported by the UNIX version of **wimcapture** on UNIX filesystems (it's not yet implemented for Windows filesystems).

Windows also treats certain types of reparse points specially. For example, Windows applications reading from deduplicated, WIM-backed, or system-compressed files always see the dereferenced data, even if they ask not to. Therefore, **wimcapture** on Windows will store these files dereferenced, not as reparse points. But **wimcapture** on UNIX in NTFS-3G mode cannot dereference these files and will store them as reparse points instead. This difference can be significant in certain situations, e.g. when capturing deduplicated files which, to be readable after extraction, require that the chunk store also be present.

OPTIONS

--boot

Mark the new image as the "bootable" image of the WIM. The "bootable" image is the image which the Windows bootloader will use when loading Windows PE from the WIM.

--check

Include extra integrity information in the resulting WIM. With **wimappend**, also check the integrity of the WIM before appending to it. Also verify the integrity of any WIMs specified by **--update-of** and/or **--delta-from**.

--include-integrity

Include extra integrity information in the resulting WIM, i.e. like **--check** but don't do any verification beforehand.

--compress=*TYPE*[:*LEVEL*]

With **wimcapture**, use the specified compression format in the new WIM file. *TYPE* may be "none", "XPRESS" (alias: "fast"), "LZX" (alias: "maximum"), or "LZMS" (alias: "recovery"). *TYPE* is matched case-insensitively. The default is "LZX".

You can optionally also specify an integer compression *LEVEL*. The compression level specifies how hard the compression algorithm for the specified compression *TYPE* will work to compress the data. The values are scaled so that 20 is quick compression, 50 is medium compression, and 100 is high compression. However, you can choose any value and not just these particular values. The default is 50.

This option only affects the compression type used in non-solid WIM resources. If you are creating a solid WIM (using the **--solid** option), then you probably want **--solid-compress** instead.

Be careful if you choose LZMS compression. It is not compatible with wimlib before v1.6.0, WIMGAPI before Windows 8, DISM before Windows 8.1, and 7-Zip before v15.12. Also note that choosing LZMS compression does not automatically imply solid-mode compression, as it does with DISM. Use **--solid** if you want to create a solid WIM, or "ESD file".

--chunk-size=*SIZE*

With **wimcapture**, use a compression chunk size of *SIZE* bytes. A larger compression chunk size results in a better compression ratio. wimlib supports different chunk sizes depending on the compression type:

- XPRESS: 4K, 8K, 16K, 32K, 64K

- LZX: 32K, 64K, 128K, 256K, 512K, 1M, 2M
- LZMS: 32K, 64K, 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, 64M, 128M, 256M, 512M, 1G

You can provide the full number (e.g. 32768), or you can use one of the K, M, or G suffixes. KiB, MiB, and GiB are also accepted.

This option only affects the chunk size used in non-solid WIM resources. If you are creating a solid WIM (using the **--solid** option), then you probably want **--solid-chunk-size** instead.

Use this option with caution if compatibility with Microsoft's WIM software is desired, since their software has limited support for non-default chunk sizes.

--solid
With **wimcapture**, create a "solid" WIM file that compresses files together rather than independently. This results in a significantly better compression ratio, but it comes at the cost of slow compression with very high memory usage, reduced compatibility, and slow random access to the resulting WIM file.

By default this enables solid LZMS compression, thereby creating a file equivalent to one created with DISM's **/compress**:*recovery* option. Such files are also called "ESD files" and were first supported by WIMGAPI in Windows 8, by DISM in Windows 8.1, and by 7-Zip 15.12.

--solid-compress=*TYPE*[:*LEVEL*]
Like **--compress**, but set the compression type used in solid resources. The default is LZMS compression. This option only has an effect when **--solid** is also specified.

--solid-chunk-size=*SIZE*
Like **--chunk-size**, but set the chunk size used in solid resources. The default, assuming LZMS compression, is 64MiB (67108864); this requires about 640MiB of memory per thread. This option only has an effect when **--solid** is also specified. Note: Microsoft's WIM software is not compatible with LZMS chunk sizes larger than 64MiB.

--threads=*NUM_THREADS*
Number of threads to use for compressing data. Default: autodetect (number of available CPUs).

--rebuild
With **wimappend**, rebuild the entire WIM rather than appending the new data to the end of it. Rebuilding the WIM is slower, but will save some space that would otherwise be left as a hole in the WIM. Also see **wimoptimize**(1).

--flags=*EDITIONID*
Specify a string to use in the <FLAGS> element of the XML data for the new image.

--image-property *NAME=VALUE*
Assign an arbitrary property to the new image in the XML document of the WIM. *VALUE* is the string to set as the property value. *NAME* is the name of the image property, for example "NAME", "DESCRIPTION", or "TOTALBYTES". The name can contain forward slashes to indicate a nested XML element; for example, "WINDOWS/VERSION/BUILD" indicates the BUILD element nested within the VERSION element nested within the WINDOWS element. A bracketed number can be used to indicate one of several identically-named elements; for example, "WINDOWS/LANGUAGES/LANGUAGE[2]" indicates the second "LANGUAGE" element nested within the "WINDOWS/LANGUAGES" element. When adding a list of elements in this way, they must be specified in sequential order. Note that element names are case-sensitive. This option may be specified multiple times.

--dereference
(UNIX-like systems only) Follow symbolic links and archive the files they point to, rather than archiving the links themselves.

--config=*FILE*

Specifies a configuration file (UTF-8 or UTF-16LE encoded; plain ASCII also works) for capturing the new image. The configuration file specifies files that are to be treated specially during the image capture.

The format of the configuration file is INI-style; that is, it is arranged in bracketed sections. Currently, the following sections are recognized:

- [ExclusionList] --- contains a list of path globs to exclude from capture. If a directory is matched, both the directory and its contents are excluded.
- [ExclusionException] --- contains a list of path globs to include, even when the file or directory also matches a glob in [ExclusionList]. If a directory is matched, then all its contents are included as well. Files or directories *within* a directory excluded by [ExclusionList] may even be included using this, though currently it only works for absolute globs (those that begin with "/" or "\"); for example, "/dir/file" can be included while "/dir" can be excluded, but including simply "file" won't work in that case.
- [PrepopulateList] --- this does not affect capture, but if the image is applied later with **--wimboot**, these are globs of files that shall be extracted normally, not as WIMBoot "pointer files". If a directory is matched, all files and subdirectories are also matched recursively.

Path globs may contain the '*' and '?' meta-characters. Relative globs (e.g. *.mp3) match against a filename in any directory. Absolute globs (e.g. /dir/file), are treated as paths starting at the main directory being captured, or the root of the NTFS volume for NTFS volume capture mode. Do not use drive letters in the paths; they will be ignored. Path separators may be either forwards slashes or backwards slashes.

Lines beginning with the '#' or ';' characters are treated as comments and ignored. Globs with whitespace in them need not be quoted; however, if they are, both double and single quotes are accepted.

If this option is not specified the following default configuration file is used:

```
[ExclusionList]
\$ntfs.log
\hiberfil.sys
\pagefile.sys
\swapfile.sys
\System Volume Information
\RECYCLER
\$RECYCLE.BIN
\$Recycle.Bin
\Windows\CSC
```

However, special behavior applies if **--wimboot** is also specified. By default, with **--wimboot** specified, the file Windows/System32/WimBootCompress.ini in the directory being captured will be used as the configuration file. However, this can be overridden using **--config**; and this also causes the specified configuration file to be saved in the WIM image as Windows/System32/WimBootCompress.ini, overriding any that may be present on the filesystem.

--unix-data

(UNIX-like systems only) Store UNIX-specific metadata and special files. This includes: standard UNIX file permissions (owner, group, and mode); device nodes, named pipes, and sockets; and extended attributes (Linux only). This information can later be restored by **wimapply** with the **--unix-data** option.

UNIX-specific information is ignored by Microsoft's WIM software and by the Windows version of wimlib.

--no-acls
Do not capture files' security descriptors.

--strict-acls
Fail immediately if the full security descriptor of any file cannot be read. On Windows, the default behavior without this option is to first try omitting the SACL from the security descriptor, then to try omitting the security descriptor entirely. The purpose of this is to capture as much data as possible without always requiring Administrator privileges. However, if you desire that all security descriptors be captured exactly, you may wish to provide this option, although the Administrator should have permission to read everything anyway.

--rpfix, **--norpfix**
Set whether to fix targets of absolute symbolic links (reparse points in Windows terminology) or not. When enabled (**--rpfix**), absolute symbolic links that point inside the directory tree being captured will be adjusted to be absolute relative to the root of the directory tree being captured. When disabled (**--norpfix**), absolute symbolic links will be captured exactly as is.

The default behavior of **wimcapture** is equivalent to **--rpfix**. The default behavior of **wimappend** is equivalent to **--rpfix** if reparse point fixups have previously been done on *WIMFILE*, otherwise **--norpfix**.

In the case of a multi-source capture, (**--source-list** specified), passing **--norpfix** is recommended. Otherwise, reparse point fixups will be disabled on all capture sources destined for non-root locations in the WIM image, while capture sources destined for the WIM root will get the default behavior from the previous paragraph.

--source-list
wimcapture and **wimappend** support creating a WIM image from multiple separate files or directories. When **--source-list** is specified, the *SOURCE* argument specifies the name of a text file, each line of which is either 1 or 2 whitespace separated file paths. The first file path, the source, specifies the path to a file or directory to capture into the WIM image. It may be either absolute or relative to the current working directory. The second file path, if provided, is the target and specifies the path in the WIM image that this file or directory will be saved as. Leading and trailing slashes in the target are ignored, except if it consists entirely of slashes (e.g. "/"), which indicates that the directory is to become the root of the WIM image. If omitted, the target string defaults to the same as the source string.

An example source list file is as follows:

```
# Make the WIM image from the 'winpe' directory
winpe    /

# Send the 'overlay' directory to '/overlay' in the WIM image
overlay  /overlay

# Overlay a separate directory directly on the root of the WIM image.
/data/stuff          /
```

Subdirectories in the WIM are created as needed. Multiple source directories may share the same target, which implies an overlay. In the event that this results a nondirectory file being added to the WIM image multiple times, the last version (as listed in the source list file) overrides any earlier version.

File paths containing whitespace may be quoted with either single quotes or double quotes. Quotes may not be escaped.

Lines consisting only of whitespace and lines beginning with '#' preceded by optional whitespace are ignored.

As a special case, if *SOURCE* is "-", the source list is read from standard input rather than an external file.

The NTFS volume capture mode on UNIX-like systems cannot be used with **--source-list**, as only capturing a full NTFS volume is supported.

--pipable

With **wimcapture**, create a wimlib-specific "pipable" WIM which can be captured and applied fully sequentially. If *WIMFILE* is specified as "-", then the pipable WIM is written directly to standard output; otherwise, it is written to disk as usual. The image in the pipable WIM can be later be applied with **wimapply**, either from disk or from standard input. A typical use of pipable WIMs might involve streaming the WIM image to a remote server when capturing it and/or streaming the WIM image from a remote server when applying it.

Generally, all the **wimlib-imagex** commands work on both pipable and non-pipable WIMs. **wimoptimize** and **wimexport** may also be used to convert between pipable WIMs and non-pipable WIMs. However, there are a few limitations of pipable WIMs:

- Pipable WIMs are a wimlib extension which are *not* compatible with Microsoft's WIM software or with other programs such as 7-Zip.
- Using **wimappend**, multiple images may be added to a pipable WIM. This is supported, though it is less efficient than doing so with non-pipable WIMs because a pipable WIM is fully rebuilt each time it is appended to; and when piping such a WIM to **wimapply** to extract an image, some unneeded data will be sent over the pipe.
- Although a pipable WIM image may be updated using **wimupdate**, it requires a full rebuild of the WIM file, making it less efficient than updating a non-pipable WIM.
- Solid pipable WIMs are not yet supported.

--not-pipable

With **wimappend**, rebuild the WIM file in the non-pipable (regular) format. This option is only useful if you happen to be adding an image to a pipable WIM (see **--pipable**) which you want in non-pipable format instead. Note that **wimoptimize**(1) can also be used to convert between non-pipable and pipable WIMs.

--update-of=[*WIMFILE*:]*IMAGE*

Hint that the image being captured or appended from *SOURCE* is mostly the same as the existing image *IMAGE* in *WIMFILE*, but captured at a later point in time, possibly with some modifications in the intervening time. This is designed to be used in incremental backups of the same filesystem or directory tree. *IMAGE* can be a 1-based index or name of an existing image in *WIMFILE*. It can also be a negative integer to index backwards into the images (e.g. -1 means the last existing image in *WIMFILE*).

When this option is provided, the capture or append of the new image will be optimized by not reading files that, based on metadata such as timestamps, appear not to have been modified since they were archived in the existing *IMAGE*. Barring manipulation of timestamps, this option only affects performance and does not change the resulting WIM image (but see note below).

As shown, the full syntax for the argument to this option is to specify the WIM file, a colon, and the image; for example, "--update-of mywim.wim:1". However, the WIM file and colon may be omitted if **--delta-from** is specified exactly once, in which case the WIM defaults to that specified in **--delta-from**, or if the operation is **wimappend** rather **wimcapture**, in which case the WIM defaults to the one being appended to.

Note: in the Windows version of wimlib, it has been observed that **--update-of** mode is not completely reliable at detecting changes in file contents, sometimes causing the old contents of a few files to be archived rather than the current contents. The cause of this problem is that Windows does not immediately update a file's last modification timestamp after every write to that file. Unfortunately, there is no known way for applications like wimlib to automatically work around this bug. Manual workarounds are possible; theoretically, taking any action that causes the problematic files

to be closed, such as restarting applications or the computer itself, should cause the files' last modification timestamps to be updated. Also note that wimlib compares file sizes as well as timestamps in determining whether a file has changed, which helps make the problem less likely to occur; and the problem does not occur on other operating systems such as Linux which maintain files' last modification timestamps correctly.

--delta-from=*WIMFILE*

Capture or append the new image as a "delta" from *WIMFILE*. Any file data that would ordinarily need to be archived in the new or updated WIM is omitted if it is already present in the *WIMFILE* on which the delta is being based. The resulting WIM will still contain a full copy of the image metadata, but this is typically only a small fraction of a WIM's total size.

This option can be specified multiple times, in which case the resulting delta WIM will only contain file data not present in any of the specified base WIMs.

To operate on the resulting delta WIM using other commands such as **wimapply**, you must specify the delta WIM as the WIM file to operate on, but also reference the base WIM(s) using the **--ref** option. Beware: to retain the proper functioning of the delta WIM, you can only add, not delete, files and images to the base WIM(s) following the capture of a delta from it.

--delta-from may be combined with **--update-of** to increase the speed of capturing a delta WIM.

As an example, consider the following backup and restore sequence:

(initial backup)

$ wimcapture /some/directory bkup-base.wim

(some days later, create second backup as delta from first)

$ wimcapture /some/directory bkup-2013-08-20.dwm \
    --update-of bkup-base.wim:-1 --delta-from bkup-base.wim

(restoring the second backup)

$ wimapply bkup-2013-08-20.dwm --ref=bkup-base.wim 1 \
    /some/directory

However, note that as an alternative to the above sequence that used a delta WIM, the second backup could have simply been appended to the WIM as new image using **wimappend**. Delta WIMs should be used only if it's desired to base the backups or images on a separate, large file that is rarely modified.

--delta-from is supported by both **wimcapture** and **wimappend**.

Delta WIMs are compatible with Microsoft's WIM software. For example, you can use the /ref option of ImageX to reference the base WIM(s), similar to above.

Additional note: wimlib is generalized enough that you can in fact combine **--pipable** and **--delta-from** to create pipable delta WIMs. In such cases, the base WIM(s) must be captured as pipable as well as the delta WIM, and when applying an image, the base WIM(s) must be sent over the pipe after the delta WIM.

--wimboot

Mark the image as WIMBoot-compatible. See Microsoft's documentation for more information about WIMBoot. With **wimcapture** this option will set the compression type to XPRESS and the chunk size to 4096 bytes; these can, however, still be overridden through the **--compress** and **--chunk-size** parameters, respectively. In addition, this option will set the configuration file to *SOURCE*\Windows\System32\WimBootCompress.ini if present and accessible; however, this may still be overridden through the **--config** parameter.

--unsafe-compact
With **wimappend**, compact the WIM archive in-place and append any new data, eliminating "holes". This is efficient, but in general this option should *not* be used because a failed or interrupted compaction will corrupt the WIM archive. For more information, see the documentation for this option to **wimoptimize**(1).

--snapshot
Create a temporary filesystem snapshot of the source directory and capture the files from it. Currently, this option is only supported on Windows, where it uses the Volume Shadow Copy Service (VSS). Using this option, you can create a consistent backup of the system volume of a running Windows system without running into problems with locked files. For the VSS snapshot to be successfully created, **wimlib-imagex** must be run as an Administrator, and it cannot be run in WoW64 mode (i.e. if Windows is 64-bit, then **wimlib-imagex** must be 64-bit as well).

--create
With **wimappend**, if the WIM file doesn't exist yet, then create it (like **wimcapture**).

NOTES

wimappend does not support appending an image to a split WIM.

Except when using **--unsafe-compact**, it is safe to abort a **wimappend** command partway through; however, after doing this, it is recommended to run **wimoptimize** to remove any data that was appended to the physical WIM file but not yet incorporated into the structure of the WIM, unless the WIM was being fully rebuilt (e.g. with **--rebuild**), in which case you should delete the temporary file left over.

wimlib-imagex creates WIMs compatible with Microsoft's software (WIMGAPI, ImageX, DISM), with some caveats:

- With **wimlib-imagex** on UNIX-like systems, it is possible to create a WIM image containing files with names differing only in case, or files with names containing the characters ':', '*', '?', '"', '<', '>', '|', or '\', which are valid on POSIX-compliant filesystems but not Windows. Be warned that such files will not be extracted by default by the Windows version of **wimlib-imagex**, and (even worse) Microsoft's ImageX can be confused by such names and quit extracting the image partway through.
- Pipable WIMs are incompatible with Microsoft's software. Pipable WIMs are created only if *WIMFILE* was specified as "-" (standard output) or if the **--pipable** flag was specified.
- WIMs captured with a non-default chunk size (with the **--chunk-size** option) or as solid archives (with the **--solid** option) or with LZMS compression (with **--compress**=LZMS or **--compress**=recovery) have varying levels of compatibility with Microsoft's software. Generally, more recent versions of Microsoft's software are more compatible.

EXAMPLES

First example: Create a new WIM 'mywim.wim' with LZX ("maximum") compression that will contain a captured image of the directory tree 'somedir'. Note that the image name need not be specified and will default to 'somedir':

```
wimcapture somedir mywim.wim
```

Next, append the image of a different directory tree to the WIM created above:

```
wimappend anotherdir mywim.wim
```

Easy enough, and the above examples of imaging directory trees work on both UNIX-like systems and Windows. Next, capture a WIM with several non-default options, including XPRESS ("fast") compression, extra integrity information, no messing with absolute symbolic links, and an image name and description:

```
wimcapture somedir mywim.wim --compress=fast \
        --check --norpfix "Some Name" "Some Description"
```

On a UNIX-like system, capture a full NTFS volume into a new WIM using the **NTFS VOLUME CAPTURE (UNIX)** mode, and name the image "Windows 7":

```
wimcapture /dev/sda2 windows7.wim "Windows 7"
```

or, on Windows, to capture a full NTFS volume you instead need to specify the root directory of the mounted volume, for example:

```
wimcapture E:\ windows7.wim "Windows 7"
```

Same as UNIX example above, but capture the WIM in the wimlib-specific "pipable" format that can be piped to **wimapply**:

```
wimcapture /dev/sda2 windows7.wim "Windows 7" --pipable
```

Same as above, but instead of writing the pipable WIM to the file "windows7.wim", write it directly to standard output through a pipe into some other program "someprog", which could, for example, be a program or script that streams the data to a server:

```
wimcapture /dev/sda2 - "Windows 7" | someprog
```

SEE ALSO

wimlib-imagex(1), **wimapply**(1) **wimoptimize**(1)