## NAME

**dd** — block-wise copy with conversions

## SYNOPSIS

**dd** [**if**=*file*] [**of**=*file*] [**bs**=*size*|[**ibs**=*size*] [**obs**=*size*]] [**skip**=*blocks*]
[**seek**=*blocks*] [**count**=*blocks*] [**status=none**|**noxfer**|**xfer**|**progress**]
[**iflag**|**oflag=append**|**direct**|**directory**|**sync**|**dsync**|**nonblock**
|**noatime**|**nocache**|**noctty**|**nofollow**[**,**...]]
[**iflag=fullblock**|**skip_bytes**|**count_bytes**[**,**...]] [**oflag=seek_bytes**]
[**conv=ucase**|**lcase**|**swab**|**sparse**|**compare**|**sync**
|**fdatasync**|**fsync**|**noerror**|**excl**|**nocreat**|**notrunc**[**,**...]]
[**conv=block**|**unblock**|**ascii**|**ebcdic**|**ibm cbs**=*size*]

## DESCRIPTION

Copies **if=** (standard input default) into **of=** (standard output default) blockwise, ignoring the first **skip=** blocks of the input and **seek=** blocks of the output (**0** default for both; bytes if **iflag=skip_bytes** or **oflag=seek_bytes**, respectively).

There are two principal modes of operation: when **bs=** is specified and none of **conv=block**|**unblock**|**lcase**|**ucase**|**swab** are, each **bs=**-sized read corresponds to a write of the same size. Otherwise, reads are **ibs=**-sized, conversions are performed, and output written in **obs=**-sized blocks.

Copy statistics (amount of full (matching the corresponding block size) and partial blocks read and written) and average speed are written to the standard error stream at the end. These statistics can also be requested at any time by sending `SIGUSR1` (`SIGINFO`) to **dd**. With **status=progress**, the speed statistics are displayed every second, unless blocked on I/O.

Unless **conv=notrunc**, the output file is truncated to its seek offset. If seeking is not available on either file, the file is read instead. If **conv=sparse** and the output file is seekable, output blocks consisting solely of null bytes are sought over instead of written. If writing to a regular file shorter than the input, this file will become sparse, and if the output ends with sought-over null bytes, truncated to the correct length. Similarly, if **conv=compare** and the output file is seekable and readable, output blocks identical to what's read from the output file will be sought over.

Copying continues until end-of-file, a read error without **conv=noerror**, a write error, or **count=** input blocks (bytes if **iflag=count_bytes**) were processed.

Conversions are applied as follows:

1. Read input block
2. If **iflag=fullblock** and read less than **ibs=**, attempt to read data until **ibs=** is matched
3. If **conv=sync** and read less than **ibs=**, expanded with null bytes (spaces if **conv=block**|**unblock**) to match **ibs=**
4. If:
   4.1. **bs=** and conditions noted above hold, block written
   4.2. Otherwise, continued
5. If **conv=swab**, each consecutive pair of bytes swapped
6. **conv=ascii**
7. **conv=ucase**|**lcase**
8. If
   8.1. **conv=block**, split into blocks on newline, space-pad or truncate to **cbs=**, and **conv=ebcdic**|**ibm**
   8.2. **conv=unblock**, read **cbs=**-sized blocks, strip trailing spaces, add newline
9. Accumulate until at least **obs=** bytes available

Each block is written in its entirety as part of one `write`(2) call, or, if that returns short, retried until exhaustion.

**conv=block|unblock** view the input as a continuous stream of bytes: they do *not* depend on the input block size, only on **cbs=**.

With **conv=fsync|fdatasync**, the output file is flushed at the end. With **iflag|oflag=nocache**, the kernel is advised that the just-read (just-written) parts of the corresponding file are not needed; this is likely to reduce the effected cache pressure.

The arguments to **bs|ibs|obs|skip|seek|count=** are in the format:
[*integer*][**cwb|KMGTPEZY**[**B**]][**x**[*integer*][**cwb|KMGTPEZY**[**B**]]]... (with at least one of {*integer*, **cwb**, **KMGTPEZY**, **B**} in each factor)
Multiple *integer*s with optional units (if the *integer* is omitted, it's taken to be **1** and the unit is required) are multiplied together when separated by **x**, and are all decimal. **KMGTPEZY** are case-insensitive. The units are as follows:

| | |
|---|---|
| **c** | *1* |
| **w** | *2* (but see **STANDARDS**) |
| **b** | *512* |
| [**KMGTPEZY**] | The relevant unit, base *1024*. |
| [**KMGTPEZY**]**B** | Likewise, base *1000*. |

For example, `3wx6bx2MB` corresponds to *3·2 · 6·512 · 2·1000^2*, and is equal to *36‘864‘000‘000* (a ridiculous value, to be sure).

## OPTIONS

**if=***file*, **of=***file*
: Use *file* instead of the standard input/output stream.

**bs=***size*
: Use *size* as the for input and output block size, and, if no incompatible conversions are specified, use a block-for-block copy algorithm (see above). This overrides **ibs=** and **obs=**.

**ibs=***size*, **obs=***size*
: Read/write in blocks of *size* instead of the default **512**.

**skip=***blocks*, **seek=***blocks*
**iflag=skip_bytes**, **oflag=seek_bytes**
: Seek over the first *blocks* blocks (or bytes) of the corresponding file. If unseekable, like a pipe — read that many blocks (bytes).

**count=***blocks*
**iflag=count_bytes**
: Stop processing after reading *blocks* blocks (or bytes).

**status=none|noxfer|xfer|progress**
: Verbosity of statistics issued to the standard error stream when finishing and receiving `SIGUSR1` (`SIGINFO`):

  **none** Nothing.
  **noxfer** Input and output block counts.
  **xfer** And the average transfer speed. This is the default.
  **progress** Also write the average and momentary transfer speeds up to once per second during the transfer.

**iflag|oflag=append|direct|directory|sync|dsync|nonblock|noatime|noctty|nofollow**
: These correspond directly to their respective `O_`*flag* `open`(2) flags, and are set when opening **if=** and **of=**, respectively, or applied to the standard input/output streams. A short summary follows.

  **append** Always write to the end of the file.
  **direct** Bypass caches. Special alignment and read sizes may be required.
  **directory** Fail to open if not a directory.
  **sync** All writes are flushed immediately.
  **dsync** The data of all writes is flushed immediately.
  **nonblock** Error instead of waiting for any length of time. May not mean anything for devices and regular files.

**noatime** Don't update access time on read.
**noctty** Don't assign controlling teletype.
**nofollow** Error if final path component is a symlink.

**iflag|oflag=nocache** Advise the kernel that the data read/written will not be used, and may hence be dropped from the cache.

**iflag=fullblock** In case of short read, continually re-try to fill a full **ibs=**-sized block.

**conv=excl** Apply `O_EXCL` to the output file creation flags: fail if it already exists.

**conv=nocreat** Remove `O_CREAT` from the output file creation flags: fail if it doesn't already exist.

**conv=notrunc** Don't truncate the output file to its seek offset.

**conv=sparse** If possible, seek over the output instead writing blocks consisting solely of null bytes.

**conv=compare** If possible, seek over the output instead overwriting blocks that would be unchanged by the write.

**conv=fdatasync|fsync** Flush the output data (and metadata) at the end.

**conv=noerror** When encountering input error, write statistics to the standard error stream, seek to next block, and continue instead of finishing. With **conv=sync**, process as if empty block, otherwise ignore.

**conv=ucase|lcase** Convert to upper (lower) case.

**conv=swab** Swap each consecutive pair of bytes. If odd amount, the final byte is unaffected.

**conv=sync** Pad input block with null bytes (spaces if **conv=block|unblock**) to **ibs=**.

**conv=block cbs=***size* Accumulate input bytes into blocks of size *size*, separated by newlines (or end of input). If a line is too long, it's truncated; the amount of truncated lines is included in the transfer statistics. If it's shorter than *size*, it's padded with spaces. The newlines are removed. Excludes **conv=unblock**.

**conv=unblock cbs=***size* Do the reverse: accumulate *size*-sized blocks from the input, strip trailing spaces, add a newline to the end. Excludes **conv=block**.

**conv=ebcdic|ibm** (implies **conv=block**)
Transliterate output into EBCDIC (or IBM EBCDIC). These exclude each other and **conv=ascii**.

**conv=ascii** (implies **conv=unblock**)
Transliterate iutput from EBCDIC (the inverse of **conv=ebcdic**). Excludes **conv=ebcdic|ibm**.

## SIGNALS

`SIGINT` Display statistics and terminate by re-raising `SIGINT`.
`SIGUSR1`, `SIGINFO` Display statistics.

## EXIT STATUS

**1** if a *file* couldn't be opened, write, read without **conv=noerror**, or **fdatasync**()/**fsync**() failed.

## SEE ALSO

For creating sparse files, try `truncate`(1); for removing the front or back off a file try `tail`(1) or `head`(1)/`truncate`(1), respectively — they will be drastically faster.

`fcntl`(2), `fsync`(2), `open`(2), `posix_fadvise`(2), `toupper`(3)

## STANDARDS

Conforms to IEEE Std 1003.1-2024 ("POSIX.1"); the `SIGUSR1` (`SIGINFO`) behaviour is implementation-defined — this behaviour is compatible with 4.4BSD; using `SIGUSR1` where unavailable is compatible with the GNU system. This issue of the standard adds **iflags=fullblock**; this is a 13-year-old

mistake, because they were actually trying to standardise **iflag=fullblock**, which is implemented by the GNU system, FreeBSD, and the illumos gate, and only I noticed they called it the wrong thing, apparently. This implementation's **iflag=fullblock** is compatible with the standard's **iflags=fullblock**.

IEEE Std 1003.1-2024 ("POSIX.1") doesn't specify **iflag=** except for **fullblock**, **oflag=**, **status=**, nor **conv=excl|fdatasync|fsync|nocreat|sparse**. NetBSD allows directly-O_-mappable flags. FreeBSD just **iflag=fullblock|direct** and **oflag=fsync|sync|direct** (with **sync** equivalent to **fsync**). Both support **conv=sparse**. Both and OpenBSD support **conv=fsync**.

**status=** is an extension, also present on the GNU system and FreeBSD (although without **status=xfer**). OpenBSD supports **status=noxfer|none** only.

**conv=compare** is an extension, mimicking the e2image(8) **-c** flag.

The standard also doesn't mandate that **skip=**, **seek=**, or **count=** take the same number format as the **bs=** family, but all known implementations do. However, the standard format is a simplified

*integer*[**kb**][**x***integer*[**kb**]] . . .

This implementation extends the format available on the GNU system, which doesn't allow **B** without a unit and lower-case **MGTPEZY**. **w** is nominally the word (*int*) size. Version 5 AT&T UNIX and AT&T System III UNIX hard-code *2*, but 3BSD uses the actual size of an *int*; as all platforms capable of rendering this document have 32-bit *int*s, **w** is *4* on the BSD.

## HISTORY

Appeared in Version 5 AT&T UNIX as dd(I):

dd – convert and copy a file

Supporting **if=**, **of=**, **ibs=**, **obs=**, **bs=**, **cbs=**, **skip=**, **count=**, and **conv=ascii|ebcdic|lcase|ucase|swab|noerror|sync**[**,** . . .], with numbers as

*integer*[**kbw**][**x**|∗*integer*[**kbw**]] . . .

for *1024*, *512*, and *2*, respectively.

This could be considered fully-formed: every semantic is as in modern day. SIGINT simply exits instead of re-raising. **BUGS** in Version 6 AT&T UNIX note that the (un)blocking should be forked out of **conv=ascii|ebcdic**. The only significant difference is the conversion tables are different than the ones mandated by POSIX, which, rather prophetically, is noted in the **BUGS** section:

> The ASCII↔EBCDIC conversion tables are taken from the 256 character standard in the CACM Nov, 1968. It is not clear how this relates to real life.

Indeed, it's equally unclear where IEEE Std 1003.1-2008 ("POSIX.1")'s tables originate, but judging from them being monochrome 500x GIFs, it'd be safe to go with the early Devonian.

Version 7 AT&T UNIX adds **conv=ibm**, with the same table as POSIX's, **seek=**, **files=***n* (as in "copy *n* files from (tape) input"), barreling past *n−1* end-of-files, default **1**, and changes the **w** multiplier to be the size of an *int* (still *2* on the PDP-11, but *4* on the VAX with UNIX/32V).

AT&T System III UNIX uses ASCII↔EBCDIC tables "from a proposed BTL standard April 16, 1979"; they're identical to the ones specified by POSIX, except mapping ASCII 151 to 8 rather than 11, making them not strictly reciprocal, removes **files=**, and reverts to **w** equaling *2*.

AT&T System V Release 1 UNIX uses BSIZE as the default block size and **b** multiplier; this varies depending on the default filesystem (which defaults to the "Original 512 byte file system" and *512*), but for standalone programs is equal to *1024* on the VAX and *512* elsewhere.

AT&T System V Release 2 UNIX briefly alters the output format to call the I/O units *and* truncated records "blocks".

AT&T System V Release 3 UNIX reverts this, re-gains **files=**, gains **oseek=**, equivalent to **seek=** (now a compatibility option), **iseek=**, which seeks the input instead of discarding it like **skip=**, and **conv=block|unblock**, which can be selected independently from the EBCDIC conversions. It's also rewritten to be a mess, with option parsing the length of this entire implementation.

3BSD uses the Version 7 AT&T UNIX **dd**. 4.1BSD adds **conv=block|unblock**. 4.3BSD–Reno uses the modern seek-or-read algorithm for **skip=**, and errors out if **seek=** and the output isn't seekable (rather than silently not seeking), as it still opens it write-only.

4.4BSD introduces the `SIGINFO` behaviour, also notes odd-length **conv=swab** blocks in the statistics, introduces transfer speed thereto, and adds **conv=notrunc**. **conv=ascii|ebcdic|ibm** have been replaced with tables from IEEE Std 1003.2 ("POSIX.2"), with Version 5 AT&T UNIX tables moved to **conv=oldascii|oldebcdic|oldibm**. What has somehow escaped the attention of every contemporary author and all documentation is that **conv=ibm** is *the same* as **conv=oldibm**.

4.4BSD–Lite2 adds **conv=osync**, padding output blocks with null bytes to **obs=**.