-- --

# IEEE P1003.2 Draft 11.2 – September 1991

## 4.16 dd — Convert and copy a file

### 4.16.1 Synopsis

`dd` [*operand* ... ]

### 4.16.2 Description

The `dd` utility shall copy the specified input file to the specified output file with possible conversions using specific input and output block sizes. It shall read the input one block at a time, using the specified input block size; it then shall process the block of data actually returned, which could be smaller than the requested block size. It shall apply any conversions that have been specified and write the resulting data to the output in blocks of the specified output block size. If the `bs=`*expr* operand is specified and no conversions other than `sync` or `noerror` are requested, the data returned from each input block shall be written as a separate output block; if the read returns less than a full block and the `sync` conversion is not specified, the resulting output block shall be the same size as the input block. If the `bs=`*expr* operand is not specified, or a conversion other than `sync` or `noerror` is requested, the input shall be processed and collected into full-sized output blocks until the end of the input is reached.

The processing order shall be as follows:

(1) An input block is read.

(2) If the input block is shorter than the specified input block size and the `sync` conversion is specified, null bytes shall be appended to the input 2
data up to the specified size. The remaining conversions and output shall include the pad characters as if they had been read from the input.

(3) If the `bs=`*expr* operand is specified and no conversion other than `sync` or `noerror` is requested, the resulting data shall be written to the output as a single block, and the remaining steps are omitted.

(4) If the `swab` conversion is specified, each pair of input data bytes shall be swapped. If there are an odd number of bytes in the input block, the results are unspecified.

(5) Any remaining conversions (`block`, `unblock`, `lcase`, and `ucase`) shall be performed. These conversions shall operate on the input data independently of the input blocking; an input or output fixed-length record may span block boundaries.

(6) The data resulting from input or conversion or both shall be aggregated into output blocks of the specified size. After the end of input is reached, any remaining output shall be written as a block without padding if `conv=sync` is not specified; thus the final output block may be shorter than the output block size.

### 4.16.3 Options

None.

### 4.16.4 Operands

All of the operands shall be processed before any input is read. The following operands shall be supported by the implementation:

`if=`*file* Specify the input pathname; the default is standard input.

`of=`*file* Specify the output pathname; the default is standard output. If the `seek=`*expr* conversion is not also specified, the output file shall be truncated before the copy begins, unless `conv=notrunc` is specified. If `seek=`*expr* is specified, but `conv=notrunc` is not, the effect of the copy shall be to preserve the blocks in the output file over which `dd` seeks, but no other portion of the output file shall be preserved. (If the size of the seek plus the size of the input file is less than the previous size of the output file, the output file shall be shortened by the copy.)

`ibs=`*expr* Specify the input block size, in bytes, by *expr* (default is 512).

`obs=`*expr* Specify the output block size, in bytes, by *expr* (default is 512).

`bs=`*expr* Set both input and output block sizes to *expr* bytes, superseding `ibs=` and `obs=`. If no conversion other than
`sync`, `noerror`, and `notrunc` is specified, each input block 2
shall be copied to the output as a single block without aggregating short blocks.

`cbs=`*expr* Specify the conversion block size for `block` and `unblock` in
bytes by *expr* (default is zero). If `cbs=` is omitted or given a 2
value of zero, using `block` or `unblock` produces unspecified 2
results. 2

`skip=`*n* Skip *n* input blocks (using the specified input block size) before starting to copy. On seekable files, the implementation shall read the blocks or seek past them; on nonseekable files, the blocks shall be read and the data shall be discarded.

`seek=`*n* Skip *n* blocks (using the specified output block size) from beginning of output file before copying. On nonseekable files, existing blocks shall be read and space from the current end of file to the specified offset, if any, filled with
null bytes; on seekable files, the implementation shall seek 2
to the specified offset or read the blocks as described for nonseekable files.

`count=`*n* Copy only *n* input blocks.

`conv=`*value* `[,`*value* `...]`

Where *value* s are comma-separated symbols from the following list.

`block` Treat the input as a sequence of `<newline>`-terminated or end-of-file-terminated variable length records independent of the input block boundaries. Each record shall be converted to a record with a fixed length specified by the conversion block size. Any `<newline>` shall be removed from the input line; `<space>`s shall be appended to lines that are shorter than their conversion block size to fill the block. Lines that are longer than the conversion block size shall be truncated to the largest number of characters that will fit into that size; the number of truncated lines shall be reported (see Standard Error below).

The `block` and `unblock` values are mutually exclusive.

`unblock` Convert fixed length records to variable length. Read a number of bytes equal to the conversion block size, delete all trailing `<space>`s, and append a `<newline>`.

`lcase` Map uppercase characters specified by the LC_CTYPE keyword `tolower` to the corresponding lowercase character. Characters for which no mapping is specified shall not be modified by this conversion.

The `lcase` and `ucase` symbols are mutually exclusive.

`ucase` Map lowercase characters specified by the LC_CTYPE keyword `toupper` to the corresponding uppercase character. Characters for which no mapping is specified shall not be modified by this conversion.

`swab` Swap every pair of input bytes.

`noerror` Do not stop processing on an input error. When an input error occurs, a diagnostic message shall be written on standard error, followed by the current input and output block counts in the same format as used at completion (see Standard Error). If the `sync` conversion is specified, the missing input shall be replaced with null bytes and processed normally; otherwise, the input block shall be omitted from the output.

`notrunc` Do not truncate the output file. Preserve blocks in the output file not explicitly written by this invocation of the `dd` utility. (See also the preceding `of=`*file* operand.)

`sync` Pad every input block to the size of `ibs=` buffer, appending null bytes. 2

The behavior is unspecified if operands other than `conv=` are specified more than once.

For the `bs=`, `cbs=`, `ibs=`, and `obs=` operands, the application shall supply an expression specifying a size in bytes. The expression, *expr*, can be:

(1) a positive decimal number;

(2) a positive decimal number followed by `k`, specifying multiplication by 1024;

(3) a positive decimal number followed by `b`, specifying multiplication by 512; or

(4) two or more positive decimal numbers (with or without `k` or `b`) separated by `x`, specifying the product of the indicated values.

### 4.16.5 External Influences

#### 4.16.5.1 Standard Input

If no `if=` operand is specified, the standard input shall be used. See Input Files.

#### 4.16.5.2 Input Files

The input file can be any file type.

#### 4.16.5.3 Environment Variables

The following environment variables shall affect the execution of `dd`:

**LANG** This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

**LC_ALL** This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

**LC_CTYPE** This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files), the classification of characters as upper- or lowercase, and the mapping of characters from one case to the other.

**LC_MESSAGES** This variable shall determine the language in which messages should be written.

#### 4.16.5.4 Asynchronous Events

For SIGINT, the `dd` utility shall write status information to standard error before exiting. It shall take the standard action for all other signals; see 2.11.5.4.

### 4.16.6 External Effects

#### 4.16.6.1 Standard Output

If no `of=` operand is specified, the standard output shall be used. The nature of the output depends on the operands selected.

#### 4.16.6.2 Standard Error

On completion, `dd` shall write the number of input and output blocks to standard error. In the POSIX Locale the following formats shall be used:

`"%u+%u records in\n"`, *<number of whole input blocks>*, *<number of partial input blocks>*

`"%u+%u records out\n"`, *<number of whole output blocks>*, *<number of partial output blocks>*

A partial input block is one for which *read*() returned less than the input block size. A partial output block is one that was written with fewer bytes than specified by the output block size.

In addition, when there is at least one truncated block, the number of truncated blocks shall be written to standard error. In the POSIX Locale, the format shall be:

`"%u truncated %s\n"`, *<number of truncated blocks>*, `"block"` [if *<number of truncated blocks>* is one] `"blocks"` [otherwise]

Diagnostic messages may also be written to standard error.

#### 4.16.6.3 Output Files

If the `of=` operand is used, the output shall be the same as described in Standard Output.

### 4.16.7 Extended Description

None.

### 4.16.8 Exit Status

The `dd` utility shall exit with one of the following values:

0 The input file was copied successfully.

>0 An error occurred.

### 4.16.9 Consequences of Errors

If an input error is detected and the `noerror` conversion has not been specified, any partial output block shall be written to the output file, a diagnostic message shall be written, and the copy operation shall be discontinued. If some other error is detected, a diagnostic message shall be written and the copy operation shall be discontinued.

### 4.16.10 Rationale. *(This subclause is not a part of P1003.2)*

**Examples, Usage**

The input and output block size can be specified to take advantage of raw physical I/O.

The following command:

```
dd if=/dev/rmt0h  of=/dev/rmt1h
```

copies from tape drive 0 to tape drive 1, using a common historical device naming convention.

The following command:

```
dd ibs=10  skip=1
```

strips the first 10 bytes from standard input.

A suggested implementation technique for `conv=noerror,sync` is to zero the input buffer before each read and to write the contents of the input buffer to the output even after an error. In this manner, any data transferred to the input buffer before the error was detected will be preserved. Another point is that a failed read on a regular file or a disk will generally not increment the file offset, and `dd` must then seek past the block on which the error occurred; otherwise, the input error will occur repetitively. When the input is a magnetic tape, however, the tape will normally have passed the block containing the error when the error is reported, and thus no seek is necessary.

## History of Decisions Made

**Table 4-4 – ASCII to EBCDIC Conversion**

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **0000** | 0000 | 0001 | 0002 | 0003 | 0067 | 0055 | 0056 | 0057 |
| **0010** | 0026 | 0005 | 0045 | 0013 | 0014 | 0015 | 0016 | 0017 |
| **0020** | 0020 | 0021 | 0022 | 0023 | 0074 | 0075 | 0062 | 0046 |
| **0030** | 0030 | 0031 | 0077 | 0047 | 0034 | 0035 | 0036 | 0037 |
| **0040** | 0100 | 0132 | 0177 | 0173 | 0133 | 0154 | 0120 | 0175 |
| **0050** | 0115 | 0135 | 0134 | 0116 | 0153 | 0140 | 0113 | 0141 |
| **0060** | 0360 | 0361 | 0362 | 0363 | 0364 | 0365 | 0366 | 0367 |
| **0070** | 0370 | 0371 | 0172 | 0136 | 0114 | 0176 | 0156 | 0157 |
| **0100** | 0174 | 0301 | 0302 | 0303 | 0304 | 0305 | 0306 | 0307 |
| **0110** | 0310 | 0311 | 0321 | 0322 | 0323 | 0324 | 0325 | 0326 |
| **0120** | 0327 | 0330 | 0331 | 0342 | 0343 | 0344 | 0345 | 0346 |
| **0130** | 0347 | 0350 | 0351 | 0255 | 0340 | 0275 | 0232 | 0155 |
| **0140** | 0171 | 0201 | 0202 | 0203 | 0204 | 0205 | 0206 | 0207 |
| **0150** | 0210 | 0211 | 0221 | 0222 | 0223 | 0224 | 0225 | 0226 |
| **0160** | 0227 | 0230 | 0231 | 0242 | 0243 | 0244 | 0245 | 0246 |
| **0170** | 0247 | 0250 | 0251 | 0300 | 0117 | 0320 | 0137 | 0007 |
| **0200** | 0040 | 0041 | 0042 | 0043 | 0044 | 0025 | 0006 | 0027 |
| **0210** | 0050 | 0051 | 0052 | 0053 | 0054 | 0011 | 0012 | 0033 |
| **0220** | 0060 | 0061 | 0032 | 0063 | 0064 | 0065 | 0066 | 0010 |
| **0230** | 0070 | 0071 | 0072 | 0073 | 0004 | 0024 | 0076 | 0341 |
| **0240** | 0101 | 0102 | 0103 | 0104 | 0105 | 0106 | 0107 | 0110 |
| **0250** | 0111 | 0121 | 0122 | 0123 | 0124 | 0125 | 0126 | 0127 |
| **0260** | 0130 | 0131 | 0142 | 0143 | 0144 | 0145 | 0146 | 0147 |
| **0270** | 0150 | 0151 | 0160 | 0161 | 0162 | 0163 | 0164 | 0165 |
| **0300** | 0166 | 0167 | 0170 | 0200 | 0212 | 0213 | 0214 | 0215 |
| **0310** | 0216 | 0217 | 0220 | 0152 | 0233 | 0234 | 0235 | 0236 |
| **0320** | 0237 | 0240 | 0252 | 0253 | 0254 | 0112 | 0256 | 0257 |
| **0330** | 0260 | 0261 | 0262 | 0263 | 0264 | 0265 | 0266 | 0267 |
| **0340** | 0270 | 0271 | 0272 | 0273 | 0274 | 0241 | 0276 | 0277 |
| **0350** | 0312 | 0313 | 0314 | 0315 | 0316 | 0317 | 0332 | 0333 |
| **0360** | 0334 | 0335 | 0336 | 0337 | 0352 | 0353 | 0354 | 0355 |
| **0370** | 0356 | 0357 | 0372 | 0373 | 0374 | 0375 | 0376 | 0377 |

The Options subclause is listed as ''None'' because there are no options recognized by historical dd utilities. Certainly, many of the operands could have been designed to use the Utility Syntax Guidelines, which would have resulted in the classic hyphenated option letters. In this version of this standard, dd retains its curious JCL-like syntax due to the large number of applications that depend on the historical implementation. ''Fixing'' the interface would cause an excessive compatibility problem. However, due to interest in the international community, the developers of the standard have agreed to provide an alternative syntax for

**Table 4-5 – ASCII to IBM EBCDIC Conversion**

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
|---|---|---|---|---|---|---|---|---|
| **0000** | 0000 | 0001 | 0002 | 0003 | 0067 | 0055 | 0056 | 0057 |
| **0010** | 0026 | 0005 | 0045 | 0013 | 0014 | 0015 | 0016 | 0017 |
| **0020** | 0020 | 0021 | 0022 | 0023 | 0074 | 0075 | 0062 | 0046 |
| **0030** | 0030 | 0031 | 0077 | 0047 | 0034 | 0035 | 0036 | 0037 |
| **0040** | 0100 | 0132 | 0177 | 0173 | 0133 | 0154 | 0120 | 0175 |
| **0050** | 0115 | 0135 | 0134 | 0116 | 0153 | 0140 | 0113 | 0141 |
| **0060** | 0360 | 0361 | 0362 | 0363 | 0364 | 0365 | 0366 | 0367 |
| **0070** | 0370 | 0371 | 0172 | 0136 | 0114 | 0176 | 0156 | 0157 |
| **0100** | 0174 | 0301 | 0302 | 0303 | 0304 | 0305 | 0306 | 0307 |
| **0110** | 0310 | 0311 | 0321 | 0322 | 0323 | 0324 | 0325 | 0326 |
| **0120** | 0327 | 0330 | 0331 | 0342 | 0343 | 0344 | 0345 | 0346 |
| **0130** | 0347 | 0350 | 0351 | 0255 | 0340 | 0275 | 0137 | 0155 |
| **0140** | 0171 | 0201 | 0202 | 0203 | 0204 | 0205 | 0206 | 0207 |
| **0150** | 0210 | 0211 | 0221 | 0222 | 0223 | 0224 | 0225 | 0226 |
| **0160** | 0227 | 0230 | 0231 | 0242 | 0243 | 0244 | 0245 | 0246 |
| **0170** | 0247 | 0250 | 0251 | 0300 | 0117 | 0320 | 0241 | 0007 |
| **0200** | 0040 | 0041 | 0042 | 0043 | 0044 | 0025 | 0006 | 0027 |
| **0210** | 0050 | 0051 | 0052 | 0053 | 0054 | 0011 | 0012 | 0033 |
| **0220** | 0060 | 0061 | 0032 | 0063 | 0064 | 0065 | 0066 | 0010 |
| **0230** | 0070 | 0071 | 0072 | 0073 | 0004 | 0024 | 0076 | 0341 |
| **0240** | 0101 | 0102 | 0103 | 0104 | 0105 | 0106 | 0107 | 0110 |
| **0250** | 0111 | 0121 | 0122 | 0123 | 0124 | 0125 | 0126 | 0127 |
| **0260** | 0130 | 0131 | 0142 | 0143 | 0144 | 0145 | 0146 | 0147 |
| **0270** | 0150 | 0151 | 0160 | 0161 | 0162 | 0163 | 0164 | 0165 |
| **0300** | 0166 | 0167 | 0170 | 0200 | 0212 | 0213 | 0214 | 0215 |
| **0310** | 0216 | 0217 | 0220 | 0232 | 0233 | 0234 | 0235 | 0236 |
| **0320** | 0237 | 0240 | 0252 | 0253 | 0254 | 0255 | 0256 | 0257 |
| **0330** | 0260 | 0261 | 0262 | 0263 | 0264 | 0265 | 0266 | 0267 |
| **0340** | 0270 | 0271 | 0272 | 0273 | 0274 | 0275 | 0276 | 0277 |
| **0350** | 0312 | 0313 | 0314 | 0315 | 0316 | 0317 | 0332 | 0333 |
| **0360** | 0334 | 0335 | 0336 | 0337 | 0352 | 0353 | 0354 | 0355 |
| **0370** | 0356 | 0357 | 0372 | 0373 | 0374 | 0375 | 0376 | 0377 |

the next version of this standard that conforms to the spirit of the Utility Syntax Guidelines. This new syntax will be accompanied by the existing syntax, marked as obsolescent. System implementors are encouraged to develop and promulgate a new syntax for `dd`, perhaps using a different utility name, that can be adopted for the next version of this standard.

The default `ibs=` and `obs=` sizes are specified as 512 bytes because there are existing (largely portable) scripts that assume these values. If they were left unspecified, very strange results could occur if an implementation chose an odd block size.

Historical implementations of `dd` used *creat*() when processing `of=file`. This makes the `seek=` operand unusable except on special files. More recent BSD-based implementations use *open*() (without O_TRUNC) instead of *creat*(), but fail to delete output file contents after the data copied. Since balloting showed a desire to make this behavior available, the `conv=notrunc` feature was added.

The `w` multiplier, (historically meaning *word*), is used in System V to mean 2 and in 4.2BSD to mean 4. Since *word* is inherently nonportable, its use is not supported by POSIX.2.

All references to US ASCII and to conversions to/from IBM and EBCDIC were removed in preparation for this document's acceptance by the international community. Implementations are free to have such conversions as extensions, using the `ascii`, `ibm`, and `ebcdic` keywords. However, in the interest of promoting consistency of implementation, the original material from an early draft has been restored to the rationale as an example:

> In the two tables, the conversions from ASCII to either standard EBCDIC (Table 4-4) or the IBM version of EBCDIC (Table 4-5) are shown. The differences between the two tables are underlined. In both tables, the ASCII values are the row and column headers and the EBCDIC values are found at their intersections. For example, ASCII 0012 (LF) is the second row, third column, yielding 0045 in EBCDIC. The inverted tables (for EBCDIC to ASCII conversion) are not shown, but are in one-to-one correspondence with these tables. The tables are understood to match recent System V conversion algorithms and there have been reports that earlier System V versions and the BSD version do not always conform to these; however, representatives of the BSD development group have agreed that a future version of their system will use these tables for consistency with System V.
>
> The `cbs` operand is required if any of the `ascii`, `ebcdic`, or `ibm` operands are specified. For the `ascii` operand, the input is handled as described for the `unblock` operand except that characters are converted to ASCII before the trailing `<spaces>`s are deleted. For the `ebcdic` and `ibm` operands, the input is handled as described for the `block` operand except that the characters are converted to EBCDIC or IBM EBCDIC after the trailing `<spaces>`s are added.

The `block` and `unblock` keywords are from historical BSD practice.

Early drafts only allowed two numbers separated by `x` to be used in a product when specifying `bs=`, `cbs=`, `ibs=`, and `obs=` sizes. This was changed to reflect the historical practice of allowing multiple numbers in the product as provided by Version 7 and all releases of System V and BSD.

## 4.17 `diff` — Compare two files

### 4.17.1 Synopsis

`diff [ −c | −e | −C` *n* `] [−br]` *file1 file2*

### 4.17.2 Description

The `diff` utility shall compare the contents of *file1* and *file2* and write to standard output a list of changes necessary to convert *file1* into *file2*. This list should be minimal. No output shall be produced if the files are identical.

### 4.17.3 Options

The `diff` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

`−b` Cause trailing `<blank>`s to be ignored and other strings of `<blank>`s to compare equal.

`−c` Produce output in a form that provides three lines of context.

`−C` *n* Produce output in a form that provides *n* lines of context (where *n* shall be interpreted as a positive decimal integer).

`−e` Produce output in a form suitable as input for the `ed` utility (see 4.20), which can then be used to convert *file1* into *file2*.

`−r` Apply `diff` recursively to files and directories of the same name when *file1* and *file2* are both directories.

### 4.17.4 Operands

The following operands shall be supported by the implementation:

*file1*
*file2* A pathname of a file be compared. If either the *file1* or *file2* operand is −, the standard input shall be used in its place.

If both *file1* and *file2* are directories, `diff` shall not compare block special files, character special files, or FIFO special files to any files and shall not compare regular files to directories. The system documentation shall specify the behavior of `diff` on implementation-specific file types not specified by POSIX.1 {8} when found in directories. Further details are as specified in 4.17.6.1.1.

If only one of *file1* and *file2* is a directory, `diff` shall be applied to the nondirectory file and the file contained in the directory file with a filename that is the same as the last component of the nondirectory file.