NAME

dd – convert and copy a file

SYNOPSIS

dd [*operand=value* ...]

AVAILABILITY

SUNWcsu

DESCRIPTION

dd copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block sizes may be specified to take advantage of raw physical I/O. Sizes are specified in bytes; a number may end with **k**, **b**, or **w** to specify multiplication by 1024, 512, or 2, respectively. Numbers may also be separated by **x** to indicate multiplication.

dd will read the input one block at a time, using the specified input block size; it then will process the block of data actually returned, which could be smaller than the requested block size. **dd** will apply any conversions that have been specified and write the resulting data to the output in blocks of the specified output block size.

cbs is used only if **ascii**, **unblock**, **ebcdic**, **ibm**, or **block** conversion is specified. In the first two cases, **cbs** characters are copied into the conversion buffer, any specified character mapping is done, trailing blanks are trimmed, and a new-line is added before sending the line to output. In the last three cases, characters up to new-line are read into the conversion buffer and blanks are added to make up an output record of size **cbs**. ASCII files are presumed to contain new-line characters. If **cbs** is unspecified or **0**, the **ascii**, **ebcdic**, and **ibm** options convert the character set without changing the input file's block structure; the **unblock** and **block** options become a simple file copy.

After completion, **dd** reports the number of whole and partial input and output blocks.

OPERANDS

The following operands are supported:

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

of=*file* Specify the output path; standard output is the default. If the **seek=***expr* conversion is not also specified, the output file will 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 will be to preserve the blocks in the output file over which **dd** seeks, but no other portion of the output file will 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 will be shortened by the copy.)

ibs=*n* Specify the input block size in *n* bytes (default is **512**).

obs=*n* Specify the output block size in *n* bytes (default is **512**).

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

cbs=*n* Specify the conversion block size for **block** and **unblock** in bytes by *n* (default is **0**). If **cbs=** is omitted or given a value of **0**, using **block** or **unblock** produces unspecified results.

This option is used only if ASCII or EBCDIC conversion is 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 SPACE characters 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 SPACE characters are added.

files=*n* Copy and concatenate *n* input files before terminating (makes sense only where input is a magnetic tape or similar device).

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

iseek=*n* Seek *n* blocks from beginning of input file before copying (appropriate for disk files, where **skip** can be incredibly slow).

oseek=*n* Seek *n* blocks from beginning of output file before copying.

seek=*n* Skip *n* blocks (using the specified output block size) from beginning of output file before copying. On non-seekable files, existing blocks will 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 will seek to the specified offset or read the blocks as described for non-seekable files.

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

conv=*value***[,***value*. . . **]**
Where *value*s are comma-separated symbols from the following list.

ascii Convert EBCDIC to ASCII.

ebcdic Convert ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, set up a pipeline with **dd conv=unblock** beforehand.

ibm Slightly different map of ASCII to EBCDIC. If converting fixed-length ASCII records without NEWLINEs, set up a pipeline with **dd conv=unblock** beforehand.

The **ascii**, **ebcdic**, and **ibm** values are mutually exclusive.

block Treat the input as a sequence of NEWLINE-terminated or EOF-terminated variable-length records independent of the input block boundaries. Each record is converted to a record with a fixed length specified by the conversion block size. Any NEWLINE character is removed from the input line; SPACE characters are appended to lines that are shorter than their conversion block size to fill the block. Lines that are longer than the conversion block size are truncated to the largest number of characters that will fit into that size; the number of truncated lines is reported.

unblock Convert fixed-length records to variable length. Read a number of bytes equal to the conversion block size (or the number of bytes remaining in the input, if less than the conversion block size), delete all trailing SPACE characters, and append a NEWLINE character.

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

lcase Map upper-case characters specified by the **LC_CTYPE** keyword **tolower** to the corresponding lower-case character. Characters for which no mapping is specified will not be modified by this conversion.

ucase Map lower-case characters specified by the **LC_CTYPE** keyword **toupper** to the corresponding upper-case character. Characters for which no mapping is specified will not be modified by this conversion.

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

swab Swap every pair of input bytes. If the current input record is an odd number of bytes, the last byte in the input record is ignored.

noerror Do not stop processing on an input error. When an input error occurs, a diagnostic message will be written on standard error, followed by the current input and output block counts in the same format as used at completion. If the **sync** conversion is specified, the missing input will be replaced with null bytes and processed normally; otherwise, the input block will 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 **dd**. (See also the preceding **of=***file* operand.)

sync Pad every input block to the size of the **ibs=** buffer, appending null bytes. (If either **block** or **unblock** is also specified, append SPACE characters, rather than null bytes.)

If operands other than **conv=** are specified more than once, the last specified **operand=***value* will be used.

For the **bs=**, **cbs=**, **ibs=**, and **obs=** operands, the application must 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

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

All of the operands will be processed before any input is read.

EXAMPLES

1. The following command:

   example% dd if=/dev/rmt/0h of=/dev/rmt/1h

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

2. The following command:

   example% dd ibs=10 skip=1

   strips the first 10 bytes from standard input.

3. This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the ASCII file **x** :

   example% dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase

4. Use **conv=sync** when writing to a tape as in the following example:
   example% tar cvf - . | compress | dd obs=1024k of=/dev/rmt/0 conv=sync

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **dd**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 The input file was copied successfully.

>0 An error occurred.

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

SEE ALSO

cp(1), **sed**(1), **tr**(1), **environ**(5)

DIAGNOSTICS

f+p records in(out) numbers of full and partial blocks read(written)

NOTES

Do not use **dd** to copy files between file systems having different block sizes.

Using a blocked device to copy a file will result in extra nulls being added to the file to pad the final block to the block boundary.

When **dd** reads from a pipe, using the **ibs=X** and **obs=Y** operands, the output will always be blocked in chunks of size Y. When **bs=Z** is used, the output blocks will be whatever was available to be read from the pipe at the time.

When using **dd** to copy files to a tape device, the file size must be a multiple of the device sector size (for example, 512 Kbyte). To copy files of arbitrary size to a tape device, use **tar**(1) or **cpio**(1).

For **SIGINT, dd** will write status information to standard error before exiting. It will take the standard action for all other signals.