Rocky Enterprise Linux 9.2 Manual Pages on command 'sg_dd.8'

$ man sg_dd.8

SG_DD(8)   SG3_UTILS   SG_DD(8)

NAME

    sg_dd - copy data to and from files and devices, especially SCSI devices

SYNOPSIS

    sg_dd   [bs=BS] [conv=CONV] [count=COUNT] [ibs=BS] [if=IFILE] [iflag=FLAGS] [obs=BS]

    [of=OFILE] [oflag=FLAGS] [seek=SEEK] [skip=SKIP] [--help] [--verbose] [--version]

    [blk_sgio={0|1}] [bpt=BPT] [cdbsz={6|10|12|16}] [cdl=CDL] [coe={0|1|2|3}] [coe_limit=CL]

    [dio={0|1}] [odir={0|1}] [of2=OFILE2] [retries=RETR] [sync={0|1}] [time={0|1}[,TO]] [ver?

    bose=VERB] [--dry-run] [--progress] [--verify]

DESCRIPTION

    Copy data to and from any files. Specialized for "files" that are Linux SCSI generic (sg)

    devices, raw devices or other devices that support the SG_IO ioctl (which are only found

    in the lk 2.6 series). Similar syntax and semantics to dd(1) command.

    The first group in the synopsis above are "standard" Unix dd(1) operands. The second group

    are extra options added by this utility. Both groups are defined below.

    When the --verify option is given, then the read side is the same but the on the write

    side, the WRITE SCSI command is replaced by the VERIFY SCSI command. If any VERIFY com?

    mands yields a sense key of MISCOMPARE then the verify operation will stop. The --verify

    option can only be used when OFILE is either a sg device or a block device with oflag=sgio

    also given. When the --verify option is used, this utility works in a similar fashion to

    the Unix cmp(1) command.

    This utility is only supported on Linux whereas most other utilities in the sg3_utils

    package have been ported to other operating systems. A utility called "ddpt" has similar

syntax and functionality to sg_dd. ddpt drops some Linux specific features while adding some other generic features. This allows ddpt to be ported to other operating systems.

OPTIONS

blk_sgio={0|1}

when set to 0, block devices (e.g. /dev/sda) are treated like normal files (i.e. read(2) and write(2) are used for IO). When set to 1, block devices are assumed to accept the SG_IO ioctl and SCSI commands are issued for IO. This is only supported for 2.6 series kernels. Note that ATAPI devices (e.g. cd/dvd players) use the SCSI command set but ATA disks do not (unless there is a protocol conversion as often occurs in the USB mass storage class). If the input or output device is a block de‐ vice partition (e.g. /dev/sda3) then setting this option causes the partition in‐ formation to be ignored (since access is directly to the underlying device). De‐ fault is 0. See the 'sgio' flag.

bpt=BPT

each IO transaction will be made using BPT blocks (or less if near the end of the copy). Default is 128 for logical block sizes less that 2048 bytes, otherwise the default is 32. So for bs=512 the reads and writes will each convey 64 KiB of data by default (less if near the end of the transfer or memory restrictions). When cd/dvd drives are accessed, the logical block size is typically 2048 bytes and bpt defaults to 32 which again implies 64 KiB transfers. The block layer when the blk_sgio=1 option is used has relatively low upper limits for transfer sizes (com‐ pared to sg device nodes, see /sys/block/<dev_name>/queue/max_sectors_kb).

bs=BS where BS must be the logical block size of the physical device (if either the input or output files are accessed via SCSI commands). Note that this differs from dd(1) which permits BS to be an integral multiple. Default is 512 which is usually cor‐ rect for disks but incorrect for cdroms (which normally have 2048 byte blocks). For this utility the maximum size of each individual IO operation is BS * BPT bytes.

cdbsz={6|10|12|16}

size of SCSI READ and/or WRITE commands issued on sg device names (or block devices when 'iflag=sgio' and/or 'oflag=sgio' is given). Default is 10 byte SCSI command blocks (unless calculations indicate that a 4 byte block number may be exceeded or BPT is greater than 16 bits (65535), in which case it defaults to 16 byte SCSI com‐ mands).

cdl=CDL

    allows setting of command duration limits. CDL is either a single value or two val?

    ues separated by a comma. If one value is given, it applies to both IFILE and OFILE

    (if they are pass-through devices). If two values are given, the first applies to

    IFILE while the second applies to OFILE. The value may be from 0 to 7 where 0 is

    the default and means there are no command duration limits. Command duration limits

    are only supported by 16 byte READ and WRITE commands (plus READ(32), WRITE(32) and

    the WRITE SCATTERED command, bit thay are used by this utility). If the cdbsz oper?

    and is not given and would have a value less than 16, then if CDL is greater than

    0, the cdbsz is increased to 16.

    Command duration limits can be accesses and changed in the Command duration limit A

    and B mode pages, plus the Command duration limit T2A and T2B mode pages. The sd?

    parm utility may be used to access and change these mode pages.

coe={0|1|2|3}

    set to 1 or more for continue on error ('coe'). Only applies to errors on sg de?

    vices or block devices with the 'sgio' flag set. Thus errors on other files will

    stop sg_dd. Default is 0 which implies stop on any error. See the 'coe' flag for

    more information.

coe_limit=CL

    where CL is the maximum number of consecutive bad blocks stepped over (due to

    "coe>0") on reads before the copy terminates. This only applies when IFILE is ac?

    cessed via the SG_IO ioctl. The default is 0 which is interpreted as no limit. This

    option is meant to stop the copy soon after unrecorded media is detected while

    still offering "continue on error" capability.

conv=sparse

    see the CONVERSIONS section below.

count=COUNT

    copy COUNT blocks from IFILE to OFILE. Default is the minimum (of IFILE and OFILE)

    number of blocks that sg devices report from SCSI READ CAPACITY commands or that

    block devices (or their partitions) report. Normal files are not probed for their

    size. If skip=SKIP or seek=SEEK are given and the count is derived (i.e.   not ex?

    plicitly given) then the derived count is scaled back so that the copy will not

    overrun the device. If the file name is a block device partition and COUNT is not

given then the size of the partition rather than the size of the whole device is
used. If COUNT is not given (or count=-1) and cannot be derived then an error mes?
sage is issued and no copy takes place.

dio={0|1}

default is 0 which selects indirect (buffered) IO on sg devices. Value of 1 at?
tempts direct IO which, if not available, falls back to indirect IO and notes this
at completion. If direct IO is selected and /proc/scsi/sg/allow_dio has the value
of 0 then a warning is issued (and indirect IO is performed). For finer grain con?
trol use 'iflag=dio' or 'oflag=dio'.

ibs=BS if given must be the same as BS given to 'bs=' option.

if=IFILE

read from IFILE instead of stdin. If IFILE is '-' then stdin is read. Starts read?
ing at the beginning of IFILE unless SKIP is given.

iflag=FLAGS

where FLAGS is a comma separated list of one or more flags outlined below.   These
flags are associated with IFILE and are ignored when IFILE is stdin.

obs=BS if given must be the same as BS given to 'bs=' option.

odir={0|1}

when set to one opens block devices (e.g. /dev/sda) with the O_DIRECT flag. User
memory buffers are aligned to the page size when set. The default is 0 (i.e. the
O_DIRECT flag is not used). Has no effect on sg, normal or raw files. If blk_sgio
is also set then both are honoured: block devices are opened with the O_DIRECT flag
and SCSI commands are issued via the SG_IO ioctl.

of=OFILE

write to OFILE instead of stdout. If OFILE is '-' then writes to stdout. If OFILE
is /dev/null then no actual writes are performed. If OFILE is '.' (period) then it
is treated the same way as /dev/null (this is a shorthand notation). If OFILE ex?
ists then it is _not_ truncated; it is overwritten from the start of OFILE unless
'oflag=append' or SEEK is given.

of2=OFILE2

write output to OFILE2. The default action is not to do this additional write (i.e.
when this option is not given). OFILE2 is assumed to be a normal file or a fifo
(i.e. a named pipe). OFILE2 is opened for writing, created if necessary, and closed

at the end of the transfer. If OFILE2 is a fifo (named pipe) then some other com?

mand should be consuming that data (e.g. 'md5sum OFILE2'), otherwise this utility

will block.

oflag=FLAGS

where FLAGS is a comma separated list of one or more flags outlined below.   These

flags are associated with OFILE and are ignored when OFILE is /dev/null, '.' (pe?

riod), or stdout.

retries=RETR

sometimes retries at the host are useful, for example when there is a transport er?

ror. When RETR is greater than zero then SCSI READs and WRITEs are retried on er?

ror, RETR times. Default value is zero.

seek=SEEK

start writing SEEK bs-sized blocks from the start of OFILE.   Default is block 0

(i.e. start of file).

skip=SKIP

start reading SKIP bs-sized blocks from the start of IFILE. Default is block 0

(i.e. start of file).

sync={0|1}

when 1, does SYNCHRONIZE CACHE command on OFILE at the end of the transfer. Only

active when OFILE is a sg device file name or a block device and 'blk_sgio=1' is

given.

time={0|1}[,TO]

when 1, times transfer and does throughput calculation, outputting the results (to

stderr) at completion. When 0 (default) doesn't perform timing.

If that value is followed by a comma, then TO is the command timeout in seconds for

SCSI READ, WRITE or VERIFY commands issued by this utility. The default is 60 sec?

onds.

verbose=VERB

as VERB increases so does the amount of debug output sent to stderr. Default value

is zero which yields the minimum amount of debug output. A value of 1 reports ex?

tra information that is not repetitive. A value 2 reports cdbs and responses for

SCSI commands that are not repetitive (i.e. other that READ and WRITE). Error pro?

cessing is not considered repetitive. Values of 3 and 4 yield output for all SCSI

commands (and Unix read() and write() calls) so there can be a lot of output. This

only occurs for scsi generic (sg) devices and block devices when the 'blk_sgio=1'

option is set.

-d, --dry-run

does all the command line parsing and preparation but bypasses the actual copy or

read. That preparation may include opening IFILE or OFILE to determine their

lengths. This option may be useful for testing the syntax of complex command line

invocations in advance of executing them.

-h, --help

outputs usage message and exits.

-p, --progress

this option causes a progress report to be output every two minutes until the copy

is complete. After the copy is complete a line with "completed" is printed to dis?

tinguish the final report from the prior progress reports. When used twice the

progress report is every minute, when used three times the progress report is every

30 seconds.

-v, --verbose

when used once, this is equivalent to verbose=1. When used twice (e.g. "-vv") this

is equivalent to verbose=2, etc.

-x, --verify

do a verify operation (like Unix command cmp(1)) rather than a copy. Cannot be used

with "oflag=sparse". of=OFILE must be given and OFILE must be an sg device or a

block device with "oflag=sgio" also given. Uses the SCSI VERIFY command with the

BYTCHK field set to 1. The VERIFY command is used instead of WRITE when this option

is given. There is no VERIFY(6) command. Stops on the first miscompare unless

oflag=coe is given.

-V, --version

outputs version number information and exits.

CONVERSIONS

One or more conversions can be given to the "conv=" option. If more than one is given,

they should be comma separated. sg_dd does not perform the traditional dd conversions

(e.g. ASCII to EBCDIC). Recently added conversions overlap somewhat with the flags so some

conversions are now supported by sg_dd.

nocreat

   this conversion has the same effect as "oflag=nocreat", namely: OFILE must exist,

   it will not be created.

noerror

   this conversion is very close to "iflag=coe" and is treated as such. See the "coe"

   flag. Note that an error on OFILE will stop the copy.

notrunc

   this conversion is accepted for compatibility with dd and ignored since the default

   action of this utility is not to truncate OFILE.

null   has no affect, just a placeholder.

sparse FreeBSD supports "conv=sparse" so the same syntax is supported in sg_dd. See

   "sparse" in the FLAGS sections for more information.

sync   is ignored by sg_dd. With dd it means supply zero fill (rather than skip) and is

   typically used like this "conv=noerror,sync" to have the same functionality as

   sg_dd's "iflag=coe".

FLAGS

   Here is a list of flags and their meanings:

   00   this flag is only active with iflag= and when given replaces if=IFILE. If both are

   given an error is generated. The input will be a stream of zeros, similar to using

   "if=/dev/zero" alone (but a little quicker).

   append causes the O_APPEND flag to be added to the open of OFILE. For regular files this

   will lead to data appended to the end of any existing data. Cannot be used together

   with the seek=SEEK option as they conflict. The default action of this utility is

   to overwrite any existing data from the beginning of the file or, if SEEK is given,

   starting at block SEEK. Note that attempting to 'append' to a device file (e.g. a

   disk) will usually be ignored or may cause an error to be reported.

   coe   continue on error. Only active for sg devices and block devices that have the

   'sgio' flag set. 'iflag=coe oflag=coe' and 'coe=1' are equivalent. Use this flag

   twice (e.g. 'iflag=coe,coe') to have the same action as the 'coe=2'. A medium,

   hardware or blank check error while reading will re-read blocks prior to the bad

   block, then try to recover the bad block, supplying zeros if that fails, and fi?

   nally re-read the blocks after the bad block. A medium, hardware or blank check er?

   ror while writing is noted and ignored. A miscompare sense key during a VERIFY com?

mand (i.e. --verify given) is noted and ignored when 'oflag=coe'. The recovery of the bad block when reading uses the SCSI READ LONG command if 'coe' given twice or more (also with the command line option 'coe=2'). Further, the READ LONG will set its CORRCT bit if 'coe' given thrice. SCSI disks may automatically try and remap faulty sectors (see the AWRE and ARRE in the read write error recovery mode page (the sdparm utility can access and possibly change these attributes)). Errors oc?curring on other files types will stop sg_dd. Error messages are sent to stderr. This flag is similar to 'conv=noerror,sync' in the dd(1) utility. See note about READ LONG below.

dio   request the sg device node associated with this flag does direct IO. If direct IO is not available, falls back to indirect IO and notes this at completion. If di?rect IO is selected and /proc/scsi/sg/allow_dio has the value of 0 then a warning is issued (and indirect IO is performed).

direct causes the O_DIRECT flag to be added to the open of IFILE and/or OFILE. This flag requires some memory alignment on IO. Hence user memory buffers are aligned to the page size. Has no effect on sg, normal or raw files. If 'iflag=sgio' and/or 'oflag=sgio' is also set then both are honoured: block devices are opened with the O_DIRECT flag and SCSI commands are issued via the SG_IO ioctl.

dpo   set the DPO bit (disable page out) in SCSI READ and WRITE commands. Not supported for 6 byte cdb variants of READ and WRITE. Indicates that data is unlikely to be required to stay in device (e.g. disk) cache. May speed media copy and/or cause a media copy to have less impact on other device users.

dsync causes the O_SYNC flag to be added to the open of IFILE and/or OFILE. The 'd' is prepended to lower confusion with the 'sync=0|1' option which has another action (i.e. a synchronisation to media at the end of the transfer).

excl   causes the O_EXCL flag to be added to the open of IFILE and/or OFILE.

ff   this flag is only active with iflag= and when given replaces if=IFILE. If both are given an error is generated. The input will be a stream of 0xff bytes (or all bits set).

flock after opening the associated file (i.e. IFILE and/or OFILE) an attempt is made to get an advisory exclusive lock with the flock() system call. The flock arguments are "FLOCK_EX | FLOCK_NB" which will cause the lock to be taken if available else a "temporarily unavailable" error is generated. An exit status of 90 is produced in

the latter case and no copy is done.

fua   causes the FUA (force unit access) bit to be set in SCSI READ and/or WRITE com?
      mands. This only has an effect with sg devices or block devices that have the
      'sgio' flag set. The 6 byte variants of the SCSI READ and WRITE commands do not
      support the FUA bit.

nocache

      use posix_fadvise() to advise corresponding file there is no need to fill the file
      buffer with recently read or written blocks.

nocreat

      this flag is only active in oflag=FLAGS. If present then OFILE will be opened if it
      exists. If OFILE doesn't exist then an error is generated. Without this flag a reg?
      ular (empty) file named OFILE will be created (and then filled). For production
      quality scripts where OFILE is a device node (e.g. '/dev/sdc') this flag is recom?
      mended. It guards against the remote possibility of 'dev/sdc' disappearing tempo?
      rarily (e.g. a USB memory key removed) resulting in a large regular file called
      '/dev/sdc' being created.

null   has no affect, just a placeholder.

random this flag is only active with iflag= and when given replaces if=IFILE. If both are
      given an error is generated. The input will be a stream of pseudo random bytes. The
      Linux getrandom(2) system call is used to create a seed and thereadter mrand48_r(3)
      is used to generate a pseudo random sequence, 4 bytes at a time. The quality of the
      randomness can be viewed with the ent(1) utility. This is not a high quality random
      number generator, it is built for speed, not quality. One application is checking
      the correctness of the copy and verify operations of this utility.

sgio   causes block devices to be accessed via the SG_IO ioctl rather than standard UNIX
      read() and write() commands. When the SG_IO ioctl is used the SCSI READ and WRITE
      commands are used directly to move data. sg devices always use the SG_IO ioctl.
      This flag offers finer grain control compared to the otherwise identical
      'blk_sgio=1' option.

sparse after each BS * BPT byte segment is read from the input, it is checked for being
      all zeros. If so, nothing is written to the output file unless this is the last
      segment of the transfer. This flag is only active with the oflag option. It cannot
      be used when the output is not seekable (e.g. stdout). It is ignored if the output

file is /dev/null . Note that this utility does not remove the OFILE prior to starting to write to it. Hence it may be advantageous to manually remove the OFILE if it is large prior to using oflag=sparse. The last segment is always written so regular files will show the same length and so programs like md5sum and sha1sum will generate the same value regardless of whether oflag=sparse is given or not. This option may be used when the OFILE is a raw device but is probably only useful if the device is known to contain zeros (e.g. a SCSI disk after a FORMAT command).

RETIRED OPTIONS

Here are some retired options that are still present:

append=0 | 1

when set, equivalent to 'oflag=append'. When clear the action is to overwrite the existing file (if it exists); this is the default. See the 'append' flag.

fua=0 | 1 | 2 | 3

force unit access bit. When 3, fua is set on both IFILE and OFILE; when 2, fua is set on IFILE;, when 1, fua is set on OFILE; when 0 (default), fua is cleared on both. See the 'fua' flag.

NOTES

Block devices (e.g. /dev/sda and /dev/hda) can be given for IFILE. If neither '-iflag=di?rect', 'iflag=sgio' nor 'blk_sgio=1' is given then normal block IO involving buffering and caching is performed. If only '-iflag=direct' is given then the buffering and caching is bypassed (this is applicable to both SCSI devices and ATA disks). If 'iflag=sgio' or 'blk_sgio=1' is given then the SG_IO ioctl is used on the given file causing SCSI commands to be sent to the device and that also bypasses most of the actions performed by the block layer (this is only applicable to SCSI devices, not ATA disks). The same applies for block devices given for OFILE.

Various numeric arguments (e.g. SKIP) may include multiplicative suffixes or be given in hexadecimal. See the "NUMERIC ARGUMENTS" section in the sg3_utils(8) man page.

The COUNT, SKIP and SEEK arguments can take 64 bit values (i.e. very big numbers). Other values are limited to what can fit in a signed 32 bit number.

Data usually gets to the user space in a 2 stage process: first the SCSI adapter DMAs into kernel buffers and then the sg driver copies this data into user memory (write operations reverse this sequence). This is called "indirect IO" and there is a 'dio' option to se?lect "direct IO" which will DMA directly into user memory. Due to some issues "direct IO"

is disabled in the sg driver and needs a configuration change to activate it. This is typ‐

ically done with 'echo 1 > /proc/scsi/sg/allow_dio'.

All informative, warning and error output is sent to stderr so that dd's output file can

be stdout and remain unpolluted. If no options are given, then the usage message is output

and nothing else happens.

Even if READ LONG succeeds on a "bad" block when 'coe=2' (or 'coe=3') is given, the recov‐

ered data may not be useful. There are no guarantees that the user data will appear "as

is" in the first 512 bytes.

A raw device must be bound to a block device prior to using sg_dd. See raw(8) for more

information about binding raw devices. To be safe, the sg device mapping to SCSI block de‐

vices should be checked with 'cat /proc/scsi/scsi', or sg_map before use.

Disk partition information can often be found with fdisk(8) [the "-ul" argument is useful

in this respect].

For sg devices (and block devices when blk_sgio=1 is given) this utility issues SCSI READ

and WRITE (SBC) commands which are appropriate for disks and reading from CD/DVD/HD-DVD/BD

drives. Those commands are not formatted correctly for tape devices so sg_dd should not be

used on tape devices. If the largest block address of the requested transfer exceeds a 32

bit block number (i.e 0xffff) then a warning is issued and the sg device is accessed via

SCSI READ(16) and WRITE(16) commands.

The attributes of a block device (partition) are ignored when 'blk_sgio=1' is used. Hence

the whole device is read (rather than just the second partition) by this invocation:

   sg_dd if=/dev/sdb2 blk_sgio=1 of=t bs=512

EXAMPLES

Looks quite similar in usage to dd:

   sg_dd if=/dev/sg0 of=t bs=512 count=1MB

This will copy 1 million 512 byte blocks from the device associated with /dev/sg0 (which

should have 512 byte blocks) to a file called t. Assuming /dev/sda and /dev/sg0 are the

same device then the above is equivalent to:

   dd if=/dev/sda iflag=direct of=t bs=512 count=1000000

although dd's speed may improve if bs was larger and count was suitably reduced. The use

of the 'iflag=direct' option bypasses the buffering and caching that is usually done on a

block device.

Using a raw device to do something similar on a ATA disk:

    raw /dev/raw/raw1 /dev/hda

    sg_dd if=/dev/raw/raw1 of=t bs=512 count=1MB

To copy a SCSI disk partition to an ATA disk partition:

    raw /dev/raw/raw2 /dev/hda3

    sg_dd if=/dev/sg0 skip=10123456 of=/dev/raw/raw2 bs=512

This assumes a valid partition is found on the SCSI disk at the given skip block address
(past the 5 GB point of that disk) and that the partition goes to the end of the SCSI
disk. An explicit count is probably a safer option. The partition is copied to /dev/hda3
which is an offset into the ATA disk /dev/hda . The exact number of blocks read from
/dev/sg0 are written to /dev/hda (i.e. no padding).

To time a streaming read of the first 1 GB (2 ** 30 bytes) on a disk this utility could be
used:

    sg_dd if=/dev/sg0 of=/dev/null bs=512 count=2m time=1

On completion this will output a line like: "time to transfer data was 18.779506 secs,
57.18 MB/sec". The "MB/sec" in this case is 1,000,000 bytes per second.

The 'of2=' option can be used to copy data and take a md5sum of it without needing to
re-read the data:

    mkfifo fif

    md5sum fif &

    sg_dd if=/dev/sg3 iflag=coe of=sg3.img oflag=sparse of2=fif bs=512

This will image /dev/sg3 (e.g. an unmounted disk) and place the contents in the (sparse)
file sg3.img . Without re-reading the data it will also perform a md5sum calculation on
the image.

SIGNALS

The signal handling has been borrowed from dd: SIGINT, SIGQUIT and SIGPIPE output the num?
ber of remaining blocks to be transferred and the records in + out counts; then they have
their default action. SIGUSR1 causes the same information to be output yet the copy con?
tinues. All output caused by signals is sent to stderr.

EXIT STATUS

The exit status of sg_dd is 0 when it is successful. Otherwise see the sg3_utils(8) man
page. Since this utility works at a higher level than individual commands, and there are
'coe' and 'retries' flags, individual SCSI command failures do not necessary cause the
process to exit.

An additional exit status of 90 is generated if the flock flag is given and some other process holds the advisory exclusive lock.

AUTHORS

Written by Douglas Gilbert and Peter Allworth.

REPORTING BUGS

Report bugs to <dgilbert at interlog dot com>.

COPYRIGHT

Copyright ? 2000-2021 Douglas Gilbert

This software is distributed under the GPL version 2. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SEE ALSO

cmp(1)

There is a web page discussing sg_dd at http://sg.danny.cz/sg/sg_dd.html

A POSIX threads version of this utility called sgp_dd is in the sg3_utils package. Another version from that package is called sgm_dd and it uses memory mapped IO to speed transfers from sg devices.

The lmbench package contains lmdd which is also interesting. For moving data to and from tapes see dt which is found at http://www.scsifaq.org/RMiller_Tools/index.html

To change mode parameters that effect a SCSI device's caching and error recovery see sd? parm(sdparm)

To verify the data on the media or to verify it against some other copy of the data see sg_verify(sg3_utils)

See also raw(8), dd(1), ddrescue(GNU), ddpt

sg3_utils-1.46   March 2021   SG_DD(8)