NAME

sgp_dd − copy data to and from files and devices, especially SCSI devices

SYNOPSIS

sgp_dd [*bs=BS*] [*count=COUNT*] [*ibs=BS*] [*if=IFILE*] [*iflag=FLAGS*] [*obs=BS*] [*of=OFILE*] [*oflag=FLAGS*] [*seek=SEEK*] [*skip=SKIP*] [*−−help*] [*−−version*]

[*bpt=BPT*] [*coe=*0|1] [*cdbsz=*6|10|12|16] [*deb=VERB*] [*dio=*0|1] [*sync=*0|1] [*thr=THR*] [*time=*0|1] [*verbose=VERB*] [*−−dry−run*] [*−−verbose*]

DESCRIPTION

Copy data to and from any files. Specialised for "files" that are Linux SCSI generic (sg) and raw devices. Similar syntax and semantics to **dd(1)** but does not perform any conversions. Uses POSIX threads (often called "pthreads") to increase the amount of parallelism. This improves speed in some cases.

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.

OPTIONS

bpt=*BPT*

each IO transaction will be made using *BPT* blocks (or less if near the end of the copy). Default is 128 for 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 block size is typically 2048 bytes and bpt defaults to 32 which again implies 64 KiB transfers.

bs=*BS* where *BS* **must** be the block size of the physical device. Note that this differs from **dd(1)** which permits 'bs' to be an integral multiple of the actual device block size. Default is 512 which is usually correct for disks but incorrect for cdroms (which normally have 2048 byte blocks).

cdbsz=6 | 10 | 12 | 16

size of SCSI READ and/or WRITE commands issued on sg device names. Default is 10 byte SCSI command blocks (unless calculations indicate that a 4 byte block number may be exceeded, in which case it defaults to 16 byte SCSI commands).

coe=0 | 1

set to 1 for continue on error. Only applies to errors on sg devices. Thus errors on other files will stop sgp_dd. Default is 0 which implies stop on any error. See the 'coe' flag for more information.

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 *skip=SEEK* are given and the count is deduced (i.e. not explicitly given) then that 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 and cannot be deduced then an error message is issued and no copy takes place.

deb=*VERB*

outputs debug information. If *VERB* is 0 (default) then there is minimal debug information and as *VERB* increases so does the amount of debug (max debug output when *VERB* is 9).

dio=0 | 1

default is 0 which selects indirect IO. Value of 1 attempts 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 control 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 reading 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.

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* exists then it is _not_ truncated; it is overwritten from the start of *OFILE* unless 'oflag=append' or *SEEK* is given.

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, '.' (period), or stdout.

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.

thr=*THR*
where *THR* is the number or worker threads (default 4) that attempt to copy in parallel. Minimum is 1 and maximum is 16.

time=0 | 1
when 1, the transfer is timed and throughput calculation is performed, outputting the results (to stderr) at completion. When 0 (default) no timing is performed.

verbose=*VERB*
increase verbosity. Same as *deb=VERB*. Added for compatibility with sg_dd and sgm_dd.

−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.

−v, **−−verbose**
when used once, this is equivalent to *verbose=1*. When used twice (e.g. "−vv") this is equivalent to *verbose=2*, etc.

−V, **−−version**
outputs version number information and exits.

FLAGS

Here is a list of flags and their meanings:

append causes the O_APPEND flag to be added to the open of *OFILE*. For normal 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. When given with 'iflag=', an error that is detected in a single SCSI command (typically 'bpt' blocks) is noted (by an error message sent to stderr), then zeros are substituted into the buffer for the corresponding write operation and the copy continues. Note that the **sg_dd** utility is more sophisticated in such error situations when 'iflag=coe'. When given with 'oflag=', any error reported by a SCSI WRITE command is reported to stderr and the copy continues (as if nothing went wrong).

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 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).

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.

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*.

fua causes the FUA (force unit access) bit to be set in SCSI READ and/or WRITE commands. This only has effect with sg devices. The 6 byte variants of the SCSI READ and WRITE commands do not support the FUA bit. Only active for sg device file names.

null has no affect, just a placeholder.

RETIRED OPTIONS

Here are some retired options that are still present:

coe=0 | 1

continue on error is 0 (off) by default. When it is 1, it is equivalent to 'iflag=coe oflag=coe' described in the FLAGS section above. Similar to 'conv=noerror,sync' in **dd(1)** utility. Default is 0 which implies stop on error. More advanced coe=1 processing on reads is performed by the sg_dd utility.

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

A raw device must be bound to a block device prior to using sgp_dd. See **raw(8)** for more information about binding raw devices. To be safe, the sg device mapping to SCSI block devices should be checked with 'cat /proc/scsi/scsi' before use.

Raw device partition information can often be found with **fdisk(8)** [the "−ul" argument is useful in this respect].

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 select "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.

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.

Why use sgp_dd? Because in some cases it is twice as fast as dd (mainly with sg devices, raw devices give some improvement). Another reason is that big copies fill the block device caches which has a negative impact on other machine activity.

SIGNALS

The signal handling has been borrowed from dd: SIGINT, SIGQUIT and SIGPIPE output the number 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 continues. All output caused by signals is sent to stderr.

EXAMPLES

Looks quite similar in usage to dd:

```
sgp_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 of=t bs=512 count=1000000
```

although dd's speed may improve if bs was larger and count was correspondingly scaled. Using a raw device to do something similar on a ATA disk:

```
raw /dev/raw/raw1 /dev/hda
sgp_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
sgp_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.

To do a fast copy from one SCSI disk to another one with similar geometry (stepping over errors on the source disk):

```
sgp_dd if=/dev/sg0 of=/dev/sg1 bs=512 coe=1
```

EXIT STATUS

The exit status of sgp_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.

AUTHORS

Written by Douglas Gilbert and Peter Allworth.

REPORTING BUGS

Report bugs to <dgilbert at interlog dot com>.

COPYRIGHT

SEE ALSO

A simpler, non−threaded version of this utility but with more advanced "continue on error" logic is called **sg_dd** and is also found in the sg3_utils package. The lmbench package contains **lmdd** which is also interesting. **raw(8), dd(1)**