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

$ man sg_write_long.8

SG_WRITE_LONG(8) SG3_UTILS SG_WRITE_LONG(8)

NAME

sg_write_long - send SCSI WRITE LONG command

SYNOPSIS

sg_write_long [--16] [--cor_dis] [--help] [--in=IF] [--lba=LBA]

[--pblock] [--verbose] [--version] [--wr_uncor] [--xfer_len=BTL] DEVICE

DESCRIPTION

Send the SCSI WRITE LONG (10 or 16 byte) command to DEVICE. The buffer

to be written to the DEVICE is filled with 0xff bytes or read from the

IF file. This buffer includes the logical data (e.g. 512 bytes) and the

ECC bytes.

This utility can be used to generate a MEDIUM ERROR at a specific logi?

cal block address. This can be useful for testing error handling. Prior

to such a test, the sg_dd utility could be used to copy the original

contents of the logical block address to some safe location. After the

test the sg_dd utility could be used to write back the original con?

tents of the logical block address. An alternate strategy would be to

read the "long" contents of the logical block address with sg_read_long

utility prior to testing and restore it with this utility after test?

ing.

Take care: If recoverable errors are being injected (e.g. only one or a few bits changed so that the ECC is able to correct the data) then care should be taken with the settings in the "read write error recovery" mode page. Specifically if the ARRE (for reads) and/or AWRE (for writes) are set then recovered errors will cause the lba to be reas?

signed (and the old location to be added to the grown defect list (PLIST)). This is not easily reversed and uses (one of the finite num?

ber of) the spare sectors set aside for this purpose. If in doubt it is probably safest to clear the ARRE and AWRE bits. These bits can be checked and modified with the sdparm utility. For example: "sdparm -c AWRE,ARRE /dev/sda" will clear the bits until the disk is power cycled.

In SBC-4 revision 7 all uses of SCSI WRITE LONG (10 and 16 byte) com?

mands were made obsolete apart from the case in which the WR_UNCOR bit is set. The SCSI READ LONG (10 and 16 byte) commands were made obso?

lete in the same revision.

OPTIONS

Arguments to long options are mandatory for short options as well.

-S, --16

send a SCSI WRITE LONG (16) command to DEVICE. The default ac?

tion (in the absence of this option) is to send a SCSI WRITE LONG (10) command.

-c, --cor_dis

sets the correction disabled (i.e 'COR_DIS') bit. This inhibits various other mechanisms such as automatic block reallocation, error recovery and various informational exception conditions being triggered. This bit is relatively new in SBC-3 .

-h, --help

output the usage message then exit.

-i, --in=IF

read data (binary) from file named IF and use it for the SCSI WRITE LONG command. If IF is "-" then stdin is read. If this op?

tion is not given then 0xff bytes are used as fill.

-l, --lba=LBA

where LBA is the logical block address of the sector to over? write. Defaults to lba 0 which is a dangerous block to over? write on a disk that is in use. Assumed to be in decimal unless prefixed with '0x' or has a trailing 'h'. If LBA is larger than can fit in 32 bits then the --16 option should be used.

-p, --pblock

sets the physical block (i.e 'PBLOCK') bit. This instructs DE? VICE to use the given data (unless --wr_uncor is also given) to write to the physical block specified by LBA. The default action is to write to the logical block corresponding to the given lba. This bit is relatively new in SBC-3 .

-v, --verbose

increase the degree of verbosity (debug messages).

-V, --version

output version string then exit.

-w, --wr_uncor

sets the "write uncorrected" (i.e 'WR_UNCOR') bit. This in? structs the DEVICE to flag the given lba (or the physical block that contains it if --pblock is also given) as having an unre? coverable error associated with it. Note: no data is transferred to DEVICE, other than the command (i.e. the cdb). In the absence of this option, the default action is to use the provided data or 0xff bytes (--xfer_len=BTL in length) and write it to DEVICE. This bit is relatively new in SBC-3 .

-x, --xfer_len=BTL

where BTL is the byte transfer length (default to 520). If the given value (or the default) does not match the "long" block size of the device, nothing is written to DEVICE and the appro? priate xfer_len value may be deduced from the error response which is printed (to stderr).

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

The 10 byte SCSI WRITE LONG command limits the logical block address to
a 32 bit quantity. For larger LBAs use the --16 option for the SCSI
WRITE LONG (16) command.

EXAMPLES

This section outlines setting up a block with corrupted data, checking
the error condition, then restoring useful contents to that sector.

First, if the data in a sector is important, save it with the
sg_read_long utility:

```
sg_read_long --lba=0x1234 --out=0x1234_1.img -x BTL /dev/sda
```

This utility may need to be executed several time in order to determine
what the correct value for BTL is. Next use this utility to "corrupt"
that sector. That might be done with:

```
sg_write_long --lba=0x1234 -x BTL /dev/sda
```

This will write a sector (and ECC data) of 0xff bytes. Some disks may
reject this (at least one of the author's does). Another approach is to
copy the 0x1234_1.img file (to 0x1234_2.img in this example) and change
some values with a hex editor. Then write the changed image with:

```
sg_write_long --lba=0x1234 --in=0x1234_2.img -x BTL /dev/sda
```

Yet another approach is to use the --wr_uncor option, if supported:

```
sg_write_long --lba=0x1234 --wr_uncor /dev/sda
```

Next we use the sg_dd utility to check that the sector is corrupted.
Here is an example:

```
sg_dd if=/dev/sda blk_sgio=1 skip=0x1234 of=. bs=512 count=1 ver?
bose=4
```

Notice that the "blk_sgio=1" option is given. This is to make sure that
the sector is read (and no others) and the error is fully reported.
The "blk_sgio=1" option causes the SG_IO ioctl to be used by sg_dd
rather than the block subsystem.

Finally we should restore sector 0x1234 to a non-corrupted state. A
sector full of zeros could be written with:

sg_dd if=/dev/zero of=/dev/sda blk_sgio=1 seek=0x1234 bs=512 count=1

This will result in a sector (block) with 512 bytes of 0x0 without a MEDIUM ERROR since the ECC and associated data will be regenerated and thus well formed. The 'blk_sgio=1' option is even more important in this case as it may stop the block subsystem doing a read before write (since the read will most likely fail). Another approach is to write back the original contents:

sg_write_long --lba=0x1234 --in=0x1234_1.img -x BTL /dev/sda

EXIT STATUS

The exit status of sg_write_long is 0 when it is successful. Otherwise see the sg3_utils(8) man page.

AUTHORS

Written by Saeed Bishara. Further work by Douglas Gilbert.

REPORTING BUGS

Report bugs to <dgilbert at interlog dot com>.

COPYRIGHT

SEE ALSO

sg_read_long, sg_dd (both in sg3_utils), sdparm(sdparm)