NAME

sg_write_buffer − send SCSI WRITE BUFFER commands

SYNOPSIS

sg_write_buffer [−−*bpw=CS*] [−−*dry−run*] [−−*help*] [−−*id=ID*] [−−*in=FILE*] [−−*length=LEN*] [−−*mode=MO*] [−−*offset=OFF*] [−−*read−stdin*] [−−*skip=SKIP*] [−−*specific=MS*] [−−*timeout=TO*] [−−*verbose*] [−−*version*] *DEVICE*

DESCRIPTION

Sends one or more SCSI WRITE BUFFER commands to *DEVICE*, along with data provided by the user. In some cases no data is required, or data can be read from the file given in the −−*in=FILE* option, or data is read from stdin when either −−*read−stdin* or −−*in=−* is given.

Some WRITE BUFFER command variants do not have associated data to send to the device. For example "activate_mc" activates deferred microcode that was sent via prior WRITE BUFFER commands. There is a different method used to download microcode to SES devices, see the sg_ses_microcode utility.

OPTIONS

Arguments to long options are mandatory for short options as well. The options are arranged in alphabetical order based on the long option name.

−b, **−−bpw**=*CS*

where *CS* is the chunk size in bytes. This will be the maximum number of bytes sent per WRITE BUFFER command. So if *CS* is less than the effective length then multiple WRITE BUFFER commands are sent, each taking the next chunk from the read data and increasing the buffer offset field in the WRITE BUFFER command by the appropriate amount. The default is a chunk size of 0 which is interpreted as a very large number hence only one WRITE BUFFER command will be sent. This option should only be used with modes that "download microcode, with offsets ..."; namely either mode 0x6, 0x7, 0xd or 0xe.

The number in *CS* can optionally be followed by ",act" or ",activate". In this case after WRITE BUFFER commands have been sent until the effective length is exhausted another WRITE BUFFER command with its mode set to "Activate deferred microcode mode" [mode 0xf] is sent.

−d, **−−dry−run**

Do all the command line processing and sanity checks including reading the input file. However at the point where a WRITE BUFFER SCSI command(s) would be sent, step over that call and assume it completed without errors and continue. *DEVICE* is still opened but can be /dev/null (in Unix). It is recommended to use −−*verbose* with this option to get an overview of what would have happened.

−h, **−−help**

output the usage message then exit. If used multiple times also prints the mode names and their acronyms.

−i, **−−id**=*ID*

this option sets the buffer id field in the cdb. *ID* is a value between 0 (default) and 255 inclusive.

−I, **−−in**=*FILE*

read data from file *FILE* that will be sent with the WRITE BUFFER command. If *FILE* is '−' then stdin is read until an EOF is detected (this is the same action as −−*read−stdin*). Data is read from the beginning of *FILE* except in the case when it is a regular file and the −−*skip=SKIP* option is given.

−l, **−−length**=*LEN*

where *LEN* is the length, in bytes, of data to be written to the device. If not given (and the length cannot be deduced from −−*in=FILE* or −−*read−stdin*) then defaults to zero. If the option is given and the length deduced from −−*in=FILE* or −−*read−stdin* is less (or no data is provided), then bytes of 0xff are used as fill bytes.

−m, **−−mode**=*MO*
this option sets the MODE field in the cdb. *MO* is a value between 0 (default) and 31 inclusive. Alternatively an abbreviation can be given. See the MODES section below. To list the available mode abbreviations at run time give an invalid one (e.g. '−−mode=xxx') or use the '−hh' option.

−o, **−−offset**=*OFF*
this option sets the BUFFER OFFSET field in the cdb. *OFF* is a value between 0 (default) and 2**24−1 . It is a byte offset.

−r, **−−read−stdin**
read data from stdin until an EOF is detected. This data is sent with the WRITE BUFFER command to *DEVICE*. The action of this option is the same as using '−−in=−'. Previously this option's long name was *−−raw* and it may still be used for backward compatibility.

−s, **−−skip**=*SKIP*
this option is only active when *−−in=FILE* is given and *FILE* is a regular file, rather than stdin. Data is read starting at byte offset *SKIP* to the end of file (or the amount given by *−−length=LEN*). If not given the byte offset defaults to 0 (i.e. the start of the file).

−S, **−−specific**=*MS*
MS is the MODE SPECIFIC field in the cdb. This is a 3−bit field so the values 0 to 7 are accepted. This field was introduced in SPC−4 revision 32 and can be used to specify additional events that activate deferred microcode (when *MO* is 0xD).

−t, **−−timeout**=*TO*
TO is the command timeout (in seconds) for each WRITE BUFFER command issued by this utility. Its default value is 300 seconds (5 minutes) and should only be altered if this is not sufficient.

−v, **−−verbose**
increase the level of verbosity, (i.e. debug output).

−V, **−−version**
print the version string and then exit.

MODES

Following is a list of WRITE BUFFER command settings for the MODE field. First is an acronym accepted by the *MO* argument of this utility. Following the acronym in square brackets are the corresponding decimal and hex values that may also be given for *MO*. The following are listed in numerical order.

hd [0, 0x0]
Combined header and data (obsolete in SPC−4).

vendor [1, 0x1]
Vendor specific.

data [2, 0x2]
Data (was called "Write Data" in SPC−3).

dmc [4, 0x4]
Download microcode and activate (was called "Download microcode" in SPC−3).

dmc_save [5, 0x5]
Download microcode, save, and activate (was called "Download microcode and save" in SPC−3).

dmc_offs [6, 0x6]
Download microcode with offsets and activate (was called "Download microcode with offsets" in SPC−3).

dmc_offs_save [7, 0x7]
Download microcode with offsets, save, and activate (was called "Download microcode with offsets and save" in SPC−3).

echo [10, 0xa]
: Write data to echo buffer (was called "Echo buffer" in SPC−3).

dmc_offs_ev_defer [13, 0xd]
: Download microcode with offsets, select activation events, save, and defer activate (introduced in SPC−4).

dmc_offs_defer [14, 0xe]
: Download microcode with offsets, save, and defer activate (introduced in SPC−4).

activate_mc [15, 0xf]
: Activate deferred microcode (introduced in SPC−4).

en_ex [26, 0x1A]
: Enable expander communications protocol and Echo buffer (obsolete in SPC−4).

dis_ex [27, 0x1B]
: Disable expander communications protocol (obsolete in SPC−4).

deh [28, 0x1C]
: Download application client error history (was called "Download application log" in SPC−3).

NOTES

If no *−−length=LEN* is given this utility reads up to 8 MiB of data from the given file *FILE* (or stdin). If a larger amount of data is required then the *−−length=LEN* option should be given.

The user should be aware that most operating systems have limits on the amount of data that can be sent with one SCSI command. In Linux this depends on the pass through mechanism used (e.g. block SG_IO or the sg driver) and various setting in sysfs in the Linux lk 2.6/3 series (e.g. /sys/block/sda/queue/max_sectors_kb). Devices (i.e. logical units) also typically have limits on the maximum amount of data they can handle in one command. These two limitations suggest that modes containing the word "offset" together with the *−−bpw=CS* option are required as firmware files get larger and larger. And *CS* can be quite small, for example 4096 bytes, resulting in many WRITE BUFFER commands being sent.

Attempting to download a microcode/firmware file that is too large may cause an error to occur in the pass−through layer (i.e. before the SCSI command is issued). In Linux such error reports can be obscure as in "pass through os error invalid argument". FreeBSD reports such errors well to the machine's console but returns a cryptic error message to this utility.

Downloading incorrect microcode into a device has the ability to render that device inoperable. One would hope that the device vendor verifies the data before activating it. If the SCSI WRITE BUFFER command is given values in its cdb (e.g. *LEN*) that are inappropriate (e.g. too large) then the device should respond with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD in CDB. If a WRITE BUFFER command (or a sequence of them) fails due to device vendor verification checks then it should respond with a sense key of ILLEGAL REQUEST and an additional sense code of COMMAND SEQUENCE ERROR.

All numbers given with options are assumed to be decimal. Alternatively numerical values can be given in hexadecimal preceded by either "0x" or "0X" (or has a trailing "h" or "H").

EXAMPLES

The following sends new firmware to an enclosure. Sending a 1.5 MB file in one WRITE BUFFER command caused the enclosure to lock up temporarily and did not update the firmware. Breaking the firmware file into 4 KB chunks (an educated guess) was more successful:

```
sg_write_buffer −b 4k −m dmc_offs_save −I firmware.bin /dev/sg4
```

The firmware update occurred in the following enclosure power cycle. With a modern enclosure the Extended Inquiry VPD page gives indications in which situations a firmware upgrade will take place.

EXIT STATUS

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

AUTHORS

Written by Luben Tuikov and Douglas Gilbert.

REPORTING BUGS

Report bugs to <dgilbert at interlog dot com>.

COPYRIGHT

SEE ALSO

sg_read_buffer, sg_ses_microcode(sg3_utils)