NAME

sane−test − SANE backend for testing frontends

DESCRIPTION

The **sane−test** library implements a SANE (Scanner Access Now Easy) backend that allows testing the SANE installation and SANE frontends. It provides access to a (nearly) unlimited number of virtual devices. There is no support for real scanners or cameras. However, the backend simulates scanning and setting options.

The idea is not only to find bugs in frontends but also to show all capabilities of SANE. Therefore **sane−test** implements functions and options that are not (or seldom) found in other backends.

The backend is commented out in @CONFIGDIR@/dll.conf, so either the comment character must be removed or the backend must be called explicitly. E.g. ‘scanimage −d test’ or ‘xscanimage test’.

SCAN MODE OPTIONS

Option **mode** selects the scan mode (Gray or Color).

Option **depth** determines the number of bits per sample (1. 8, or 16). Keep in mind, that this value refers to the sample, not the pixel. So depth=16 results in 48 bits per pixel in color mode. The most usual combinations are mode=Gray, depth=1 for lineart, mode=Gray, depth=8 for gray and mode=Color, depth=8 for color mode. The combination of color and 1-bit mode is quite obscure (8 colors) but allowed in the SANE standard. However, the meaning of bits is not defined. Currently 1 = high intensity and 0 = low intensity is used.

Setting option **hand−scanner** results in the test-backend behaving like a hand-scanner. Hand-scanners do not know the image height a priori. Instead, they return a height of −1. Setting this option allows one to test whether a frontend can handle this correctly. This option also enables a fixed width of 11 cm.

Setting option **three−pass** simulates a three-pass scanner. Older color scanners needed to scan the image once per color (reg/green/blue) to get the full image. Therefore, in this mode three single frames are transmitted in color mode.

Option **three−pass−order** provides support for changing the order of the three frames (see option three-pass above). A frontend should support all orders.

Option **resolution** sets the resolution of the image in dots per inch.

Option **source** can be used to simulate an Automatic Document Feeder (ADF). After 10 scans, the ADF will be "empty".

SPECIAL OPTIONS

Option **test−picture** allows one to set the image that’s returned to the frontend. While "Solid white" and "Solid black" are quite obvious, the other options need some more explanation. Color patterns are used to determine if all modes and their colors are represented correctly by the frontend. The grid should look like the same in every mode and resolution. A table of all the test pictures can be found at: http://www.meier−geinitz.de/sane/test−backend/test−pictures.html.

If option **invert−endianness** is set, the upper and lower bytes of image data in 16 bit modes are exchanged. This option can be used to test the 16 bit modes of frontends, e.g. if the frontend uses the correct endianness.

If option **read−limit** is set, the maximum amount of data transferred with each call to sane_read() is limited.

Option **read−limit−size** sets the limit for option read-limit. A low limit slows down scanning. It can be used to detect errors in frontend that occur because of wrong assumptions on the size of the buffer or timing problems.

Option **read−delay** enables delaying data to the frontend.

Option **read−delay−duration** selects the number of microseconds the backends waits after each transfer of a buffer. This option is useful to find timing-related bugs, especially if used over the network.

If option **read−return−value** is different from "Default", the selected status will be returned by every call to sane_read(). This is useful to test the frontend's handling of the SANE statuses.

If option **ppl−loss** is different from 0, it determines the number of pixels that are "lost" at the end of each line. That means, lines are padded with unused data.

Option **fuzzy−parameters** selects that fuzzy (inexact) parameters are returned as long as the scan hasn't been started. This option can be used to test if the frontend uses the parameters it got before the start of the scan (which it shouldn't).

Option **non−blocking** determines if non-blocking IO for sane_read() should be used if supported by the frontend.

If option **select−fd** is set, the backend offers a select filedescriptor for detecting if sane_read() will return data.

If option **enable−test−options** is set, a fairly big list of options for testing the various SANE option types is enabled.

Option **print−options** can be used to print a list of all options to standard error.

GEOMETRY OPTIONS

Option **tl−x** determines the top-left x position of the scan area.

Option **tl−y** determines the top-left y position of the scan area.

Option **br−x** determines the bottom-right x position of the scan area.

Option **br−y** determines the bottom-right y position of the scan area.

BOOL TEST OPTIONS

There are 6 bool test options in total. Each option is numbered. (3/6) means: this is option 3 of 6. The numbering scheme is intended for easier detection of options not displayed by the frontend (because of missing support or bugs).

Option **bool−soft−select−soft−detect** (1/6) is a bool test option that has soft select and soft detect (and advanced) capabilities. That's just a normal bool option.

Option **bool−hard−select−soft−detect** (2/6) is a bool test option that has hard select and soft detect (and advanced) capabilities. That means the option can't be set by the frontend but by the user (e.g. by pressing a button at the device).

Option **bool−hard−select** (3/6) is a bool test option that has hard select (and advanced) capabilities. That means the option can't be set by the frontend but by the user (e.g. by pressing a button at the device) and can't be read by the frontend.

Option **bool−soft−detect** (4/6) is a bool test option that has soft detect (and advanced) capabilities. That means the option is read-only.

Option **bool−soft−select−soft−detect−emulated** (5/6) is a Bool test option that has soft select, soft detect, and emulated (and advanced) capabilities.

Option **bool−soft−select−soft−detect−auto** (6/6) is a Bool test option that has soft select, soft detect, and automatic (and advanced) capabilities. This option can be automatically set by the backend.

INT TEST OPTIONS

There are 6 int test options in total.

Option **int** (1/6) is an int test option with no unit and no constraint set.

Option **int−constraint−range** (2/6) is an int test option with unit pixel and constraint range set. Minimum is 4, maximum 192, and quant is 2.

Option **int−constraint−word−list** (3/6) is an int test option with unit bits and constraint word list set.

Option **int−constraint−array** (4/6) is an int test option with unit mm and using an array without constraints.

Option **int−constraint−array−constraint−range** (5/6) is an int test option with unit mm and using an array with a range constraint. Minimum is 4, maximum 192, and quant is 2.

Option **int−constraint−array−constraint−word−list** (6/6) is an int test option with unit percent and using an array a word list constraint.

FIXED TEST OPTIONS

There are 3 fixed test options in total.

Option **fixed** (1/3) is a fixed test option with no unit and no constraint set.

Option **fixed−constraint−range** (2/3) is a fixed test option with unit microsecond and constraint range set. Minimum is −42.17, maximum 32767.9999, and quant is 2.0.

Option **fixed−constraint−word−list** (3/3) is a Fixed test option with no unit and constraint word list set.

STRING TEST OPTIONS

There are 3 string test options in total.

Option **string** (1/3) is a string test option without constraint.

Option **string−constraint−string−list** (2/3) is a string test option with string list constraint.

Option **string−constraint−long−string−list** (3/3) is a string test option with string list constraint. Contains some more entries...

BUTTON TEST OPTION

Option **button** (1/1) is a Button test option. Prints some text...

FILES

@CONFIGDIR@/test.conf

The backend configuration file (see also description of **SANE_CONFIG_DIR** below). The initial values of most of the basic SANE options can be configured in this file. A template containing all the default values is provided together with this backend. One of the more interesting values may be **number_of_devices**. It can be used to check the frontend's ability to show a long list of devices. The config values concerning resolution and geometry can be useful to test the handling of big file sizes.

@LIBDIR@/libsane−test.a

The static library implementing this backend.

@LIBDIR@/libsane−test.so

The shared library implementing this backend (present on systems that support dynamic loading).

ENVIRONMENT

SANE_CONFIG_DIR

This environment variable specifies the list of directories that may contain the configuration file. Under UNIX, the directories are separated by a colon (':'), under OS/2, they are separated by a semi-colon (';'). If this variable is not set, the configuration file is searched in two default directories: first, the current working directory (".") and then in @CONFIGDIR@. If the value of the environment variable ends with the directory separator character, then the default directories are searched after the explicitly specified directories. For example, setting **SANE_CONFIG_DIR** to "/tmp/config:" would result in directories "tmp/config", ".", and "@CONFIGDIR@" being searched (in this order).

SANE_DEBUG_TEST

If the library was compiled with debug support enabled, this environment variable controls the debug level for this backend. Higher debug levels increase the verbosity of the output.

Example: export SANE_DEBUG_TEST=4

SEE ALSO

sane(7), *http://www.meier−geinitz.de/sane/test−backend/*

AUTHOR

Henning Meier-Geinitz <henning@meier−geinitz.de>

BUGS

− config file values aren't tested for correctness