NAME

dgsh_negotiate − specify and obtain dgsh I/O file descriptors

SYNOPSIS

#include <dgsh.h>

dgsh_negotiate(int *flags*, **const char *** *program_name*,
int * *n_input_fds*, **int *** *n_output_fds*,
int ** *input_fds*, **int **** *output_fds*);

Link with *−ldgsh*.

DESCRIPTION

The **dgsh_negotiate**() function is called before a program participates in a *dgsh*(1) graph to specify the number of input and output file descriptors the program can handle, and obtain the file descriptors to be used.

The *flags* parameter adjusts the function's behavior. The following flags are defined.

DGSH_HANDLE_ERROR .
When this flag is set and the negotiation for creating the graph of communicating processes encounters an error, the function will print an error message to standard error (if required) and cause the calling program to exit with the error value *EX_PROTOCOL* (76).

The *program_name* parameter should be specified to match the name of the program calling the function, to aid error reporting and debugging.

The *n_input_fds* and *n_output_fds* parameters are used to pass by reference the number of input or output file descriptors required, and obtain upon return the corresponding number of descriptors supplied. Passing a null pointer indicates that the program can handle zero or one descriptor. In this case, if a variable contains the value of 1 when the function returns, the program can use the standard input or output for the corresponding channel and no descriptors are returned through *input_fds* or *output_fds*. Passing a value of -1 indicates that the program can handle an arbitrary number of corresponding file descriptors. In this case, upon return the variable will contain the actual number of file descriptors allocated to the program through the negotiation process.

The *input_fds* and *output_fds parameters* are used to return a pointer to a sequence of integers containing the file descriptors to use for input or output. The size of the integer sequence is equal to the returned corresponding value of *n_input_fds* and *n_output_fds*. The pointers may subsequently be used as an argument to the function *free*(3).

Each tool in the *dgsh* graph calls **dgsh_negotiate**() to take part in a peer-to-peer negotiation. A message block is circulated among tools and is filled with tools' I/O requirements. When all requirements are in place, an algorithm runs to find a solution that satisfies all requirements. If a solution is found, pipes are allocated and set up according to the solution. The appropriate file descriptors are provided to each tool and the negotiation phase ends.

RETURN VALUE

On success, the function returns 0, on failure it returns -1.

ENVIRONMENT

The following environment variables affect the negotiation to create the communication graph.

DGSH_DEBUG_LEVEL
Setting this variable to an integer (see the section **DEBUGGING** below) causes the function to output debug data regarding the negotiation on its standard error.

DGSH_DOT_DRAW
Setting this variable to a file path causes the *dgsh* negotiation to save the graph of the communication processes in that file. The graph is saved in *dot*(1) format.

DGSH_DOT_DRAW_EXIT
Setting this variable in conjunction with **DGSH_DOT_DRAW** causes all processes participating in the negotiation to exit after the graph is saved to the file.

DGSH_TIMEOUT
Setting this variable to an integer value specifies the number of seconds *dgsh* processes will wait for the negotiation to comlete before timing out and exiting. The default value is five seconds, but this value may need to be increased for negotiations that take a long time to complete.

DEBUGGING

The DGSH_DEBUG_LEVEL environment variable controls debug output, which appears in *stderr*. The default level 0 produces no debug output.

DGSH_DEBUG_LEVEL=1
Level 1 outputs the phases of the negotiation process, that is gather I/O requirements, compute the solution, and communicate the solution.

DGSH_DEBUG_LEVEL=2
Level 2 outputs the progress of the negotiation process by each command. **DGSH_DEBUG_LEVEL=3** Level 3 outputs the reading and writing of the message block by each command as it flows across the process graph.

DGSH_DEBUG_LEVEL=4
Level 4 outputs the complete debug output including the shell's initial setup of the process graph. This is very verbose.

ERROR MANAGEMENT

Things can go wrong in two ways. First, a command might exit before reaching the call to **dgsh_negotiate**() because of invalid command-line arguments for instance. Second, a command might not be able to complete the negotiation procedure because another command aborted during the negotiation. For commands that aborted before stepping into the negotiation procedure, a handler function is called on their exit and starts a negotiation procedure to inform the other commands on the *dgsh* graph of the error state. The exit handler is there for commands that link to the *dgsh* library. This happens by calling **dgsh_negotiate**() and linking to the library or with the use of *dgsh-wrap(1)*. For commands that stuck in the negotiation procedure because another command aborted during it, an alarm signal triggers an exit after 5 seconds spent in negotiation to help commands exit it.

EXAMPLES

The following simple implementation of echo does not receive any input and provides one output channel.

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>

#include "dgsh.h"

int
main(int argc, char *argv[])
{
	int n_input_fds = 0;
	int n_output_fds = 1;

	if (dgsh_negotiate(DGSH_HANDLE_ERROR, "echo", &n_input_fds,
	    &n_output_fds, NULL, NULL) != 0)
		errx(1, "Negotiation failed");

	assert(n_input_fds == 0);
	assert(n_output_fds == 1);
```

```
	++argv;
	while (*argv) {
		(void)printf("%s", *argv);
		if (*++argv)
			putchar(' ');
	}
	putchar('0);

	return 0;
}
```

The following program will enumerate its output channels.

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <err.h>
#include <unistd.h>

#include "dgsh.h"

int
main(int argc, char *argv[])
{
	int n_input_fds = 0, n_output_fds;
	int *output_fds;
	int i;

	switch (argc) {
	case 1:
		n_output_fds = -1;
		break;
	case 2:
		n_output_fds = atoi(argv[1]);
		break;
	default:
		errx(1, "usage: %s [n]", argv[0]);
	}


	if (dgsh_negotiate(DGSH_HANDLE_ERROR, argv[0], &n_input_fds,
			&n_output_fds, NULL, &output_fds) != 0)
		errx(1, "Negotiation failed");

	for (i = 0; i < n_output_fds; i++) {
		char buff[10];

		snprintf(buff, sizeof(buff), "%d0, i);
		write(output_fds[i], buff, strlen(buff));
		close(output_fds[i]);
	}

	return 0;
}
```

SEE ALSO

dgsh(1), **dgsh-wrap**(1).

AUTHOR

The **dgsh_negotiate** API and negotiation algorithm were designed by Diomidis Spinellis and extended and implemented by Marios Fragkoulis.