NAME

dgsh-wrap − allow any program to participate in an dgsh pipeline

SYNOPSIS

dgsh-wrap [**-S**] [**-i 0|a**] [**-o 0|a**] [**-eIO**] *program* [*program-arguments ...*]

#!/usr/libexec/dgsh/**dgsh-wrap -s** [**-i 0|a**] [**-o 0|a**] [**-eIO**] [*program-arguments ...*]

DESCRIPTION

dgsh-wrap takes as arguments an absolute path or the name of a program to execute and its arguments. It will participate in the *dgsh* negotiation process, and then execute the specified program connected to the negotiated input and output pipes. If the program is not specified through an absolute path, it will be executed by searching the existing path, excluding from it elements ending in *dgsh* (where programs already wrapped with *dgsh-wrap* may reside).

Arguments specified as <| are presented as additional input channels and arguments specified as >| are presented as additional output channels. Both are suitably replaced by named file descriptor paths when the command is invoked.

In the context of a *dgsh* process graph, *dgsh(1)* automatically invokes *dgsh-wrap* to allow non-dgsh compatible commands to participate in the negotiation procedure. Furthermore, the *dgsh* installation process sets up many POSIX programs wrapped with *dgsh-wrap* in order to communicate their particular I/O requirements.

OPTIONS

−e Replace <| and >| strings embedded within arguments, with names of input file descriptor paths. By default, only standalone arguments are thus replaced.

−i 0|a Specify the wrapped program's number of input channels. The **0** character specifies that the program does not read any input. The **a** character specifies that the program can read an arbitrary number of input streams; these will be automatically supplied by *dgsh-wrap* as file descriptor command line arguments.

−I Do not include the standard input to the command line arguments, when replacing <| arguments with names of input file descriptor paths. This will result in the standard input becoming available to the command as its standard input, rather than as a command line argument. Without this option, the first path with be */dev/fd/0*. When this option is given, the program will require one input channel more than those specified by the <| arguments.

−o 0|a Specify the wrapped program's number of output channels. The **0** character specifies that the program does not produce any output. The **a** character specifies that the program can write to an arbitrary number of output streams; these will be automatically supplied by *dgsh-wrap* as file descriptor command line arguments.

−O Do not include the standard output to the command line arguments, when replacing >| arguments with names of output file descriptor paths. This will result in the standard output becoming available to the command as its standard output, rather than as a command line argument. Without this option the first path with be */dev/fd/1*. When this option is given, the program will require one output channel more than those specified by the >| arguments.

−S Process flags as a shebang-invoked (*#!*) interpreter using an invocation-supplied program name. This will change the argument processing in two ways. First, it will cause the arguments being specified on the shebang line to be properly parsed as separate arguments by splitting them on

whitespace. (Most operating systems pass any of the line's arguments as a single string to the process.) Second, it will remove from the arguments the kernel-supplied path to the script that invoked *dgsh-wrap*. (The script path is not needed, because the program to wrap is specified as an argument.) If this flag is specified, it must be the first command-line argument.

−s Process flags as a shebang-invoked (*#!*) interpreter using an operating system-supplied program name. This will change the argument processing in two ways. First, it will split arguments in the shebang line, in the same way as the *-S* flag. Second, it will convert the kernel-supplied absolute path to the script that invoked *dgsh-wrap*, into a command name. If the name of the script is the same as the name of a command to be wrapped, this path can be used to derive the name of the program to execute, thus removing the need to supply the name of the program in the invocation line.

EXAMPLES

The following examples are given only to illustrate the command's functionality. Note that most of the wrappings shown here are either performed automatically or are not required, because corresponding commands with built-in *dgsh* support are already provided.

Wrap the *echo* command, specifying that it accepts no input.

```
dgsh-wrap -i 0 echo hi
```

Wrap the *cp* command, specifying that it does not perform any I/O.

```
dgsh-wrap -i 0 -o 0 cp src-file dest-file
```

Wrap the *paste* command, supplying its two arguments.

```
dgsh-wrap /usr/bin/paste "<|" "<|"
```

Wrap the *paste* command, with an invocation that processes the standard input and uses an additional input argument.

```
dgsh-wrap -I /usr/bin/paste - "<|"
```

Wrap the *paste* command, so that it will process all input channels provided to it.

```
dgsh-wrap -i a /usr/bin/paste
```

A wrapped version of the *uname* command can be created with an interpreter invocation file containing just the following line. In contrast to a shell script, no shell is ever launched.

```
#!/usr/libexec/dgsh/dgsh-wrap -S -d /bin/uname
```

Even simpler, a wrapped version of the *uname* command can be created if a) the interpreter invocation file is named *uname*, b) it resides in a directory named *dgsh* (so that it will be excluded from the subsequently used search path), and c) it contains the following line.

```
#!/usr/libexec/dgsh/dgsh-wrap -s  -d
```

SEE ALSO

dgsh(1), *dgsh-negotiate*(3)

AUTHOR

Diomidis Spinellis — <http://www.spinellis.gr>