NAME

dgsh − directed graph shell

SYNOPSIS

dgsh [*bash_options*] [command_string | file]

DESCRIPTION

dgsh is a modified version of *bash* that allows the specification of pipelines with non-linear non-uniform operations. These form a directed acyclic process graph, which is typically executed by multiple processor cores, thus increasing the operation's processing throughput. The *dgsh* command is equivalent to invoking the modified version of *bash* with the *--dgsh* argument in order to enable the *dgsh*-specific inter-process communication functionality.

INTER-PROCESS COMMUNICATION

Dgsh provides three new ways for expressing inter-process communication.

Multipipes are expressed as usual Unix pipelines, but can connect commands with more than one output or input channel. As an example, the *comm* command supplied with *dgsh* expects two input channels and produces on its output three output channels: the lines appearing only in first (sorted) channel, the lines appearing only in the second channel, and the lines appearing in both. Connecting the output of the *comm* command to the *cat* command supplied with *dgsh* will make the three outputs appear in sequence, while connecting it to the *paste* command supplied with *dgsh* will make the output appear in its customary format. *Dgsh* handles the following programs as being multipipe compatible: a) those that are linked with the *dgsh* library; b) scripts that include in their first line one of the strings `dgsh-wrap`, `env dgsh`, or `--dgsh`; c) scripts whose second line starts with `#!dgsh`.

Multipipe blocks are enclosed within double braces: {{ ... }}. These a) send the input received on their input side to the asynchronously-running processes that reside within the block, and, b) pass the output produced by the processes within the block to their output side. Multipipe blocks typically receive input from more than one channel and produce more than one output channel. For example, a multipipe block that runs *md5sum* and *wc -c* receives two inputs and produces two outputs: the MD5 hash of its input and the input's size. Data to multipipe blocks are typically provided with an *dgsh*-aware version of *tee* and collected by *dgsh*-aware versions of programs such as *cat* and *paste.*

Stored values offer a convenient way for communicating computed values between arbitrary processes on the graph. They allow the storage of a data stream's last record into a named buffer. This record can be later retrieved asynchronously by one or more readers. Data in a stored value can be piped into a process or out of it, or it can be read using the shell's command output substitution syntax. Stored values are implemented internally through Unix-domain sockets, a background-running store program, *dgsh-writeval*, and a reader program, *dgsh-readval.* The behavior of a stored value's IO can be modified by adding flags to *dgsh-writeval* and *dgsh-readval.*

SYNTAX EXTENSIONS

The syntax of *bash* is extended by *dgsh* as follows.

```
<dgsh_block>      ::= '{{' <dgsh_list> '}}'

<dgsh_list>       ::= <dgsh_list_item> '&'
                      <dgsh_list_item> <dgsh_list>

<dgsh_list_item> ::= <simple_command>
                      <dgsh_block>
                      <dgsh_list_item> '|' <dgsh_list_item>
```

EXAMPLES

Report file type, length, and compression performance for a URL retrieved from the web. The web file never touches the disk.

```
#!/usr/bin/env dgsh

curl -s "$1" |
tee |
{{
       echo -n 'File type:' &
       file - &

       echo -n 'Original size:' &
       wc -c &

       echo -n 'xz:' &
       xz -c | wc -c &

       echo -n 'bzip2:' &
       bzip2 -c | wc -c &

       echo -n 'gzip:' &
       gzip -c | wc -c &
}} |
cat
```

List the names of duplicate files in the specified directory

```
#!/usr/bin/env dgsh

# Create list of files
find "$@" -type f |

# Produce lines of the form
# MD5(filename)= 811bfd4b5974f39e986ddc037e1899e7
xargs openssl md5 |

# Convert each line into a "filename md5sum" pair
sed 's/^MD5(//;s/)= / /' |

# Sort by MD5 sum
sort -k2 |

tee |
{{
       # Print an MD5 sum for each file that appears more than once
       awk '{print $2}' | uniq -d &

       # Promote the stream to gather it
       cat &
}} |
# Join the repeated MD5 sums with the corresponding file names
# Join expects two inputs, second will come from scatter
join -2 2 |
```

Check if the script is running under *dgsh* or regular *bash* (for polyglot scripts)

```
if {{ : ; }} ; then
    echo dgsh
else
    echo bash
fi 2>/dev/null
```

SEE ALSO

dgsh-tee(1), **dgsh-wrap**(1), **dgsh-writeval**(1), **dgsh-readval**(1), **dgsh-monitor**(1) **dgsh-conc**(1), **dgsh-httpval**(1), **dgsh-merge-sum**(1)

AUTHOR

Dgsh was designed by Diomidis Spinellis — <http://www.spinellis.gr> — and implemented by Marios Fragkoulis. The current design and capabilities of *dgsh* have been significantly influenced by amazing feedback generously provided by Doug McIlroy.

BUGS

Report bugs through https://github.com/dspinellis/dgsh/issues.