NAME

`mrkey` – print key (column-1) values from `mrskew` output

SYNOPSIS

`mrkey` [`--delimiter=`*string* | `-d` *string*] [`--help` | `-?`] [`--license`] [`--man`] [`--newline` | `-n`] [`--rows=`*range_list* | `-r` *range_list*] [`--version`] [*file*]

DESCRIPTION

`mrkey` reads `mrskew` output and prints the value in column 1 for the rows you specify. `mrkey` defines column 1 as everything beneath the first sequence of dashes that it finds on line 2 of its input. Thus, to use `mrkey`, you must use the `mrskew --dashes` option (which is the default). If no input *file* is specified, `mrkey` takes its input from the standard input device.

`mrkey` makes it easy to use the output from one `mrskew` report as input into another.

OPTIONS

`--delimiter=`*string*, `-d` *string* Use *string* as the delimiter between keys when there are multiple keys in the output. Default value is a newline character.

`--help`, `-?` Print usage information and exit.

`--license` Print license key information and exit.

`--man` Print the manual page and exit.

`--newline`, `-n` Terminate the `mrkey` output with a newline character. Default is `--nonewline`.

`--rows=`*range-list*, `-r` *range-list* Print the key (that is, column-1) values from the rows specified in *range-list*. The list may contain positive integers (like `1`), closed ranges of positive integers (like `2-5`), or open ranges of positive integers (like `-5` or `1-`), separated by commas (like `1,11-15`). The default is `--rows=1`. To print all key values use `--rows=1-`.

`--version` Print the version number and exit.

EXAMPLES

Given the following `mrskew` output:

```
$ cat profile.txt
CALL-NAME                    DURATION       %  CALLS      MEAN       MIN       MAX
---------------------------  --------  ------  -----  --------  --------  --------
FETCH                        0.000004   28.6%      1  0.000004  0.000004  0.000004
db file sequential read      0.000003   21.4%      1  0.000003  0.000003  0.000003
SQL*Net message to client    0.000002   14.3%      1  0.000002  0.000002  0.000002
SQL*Net message from client  0.000002   14.3%      1  0.000002  0.000002  0.000002
EXEC                         0.000002   14.3%      1  0.000002  0.000002  0.000002
PARSE                        0.000001    7.1%      1  0.000001  0.000001  0.000001
---------------------------  --------  ------  -----  --------  --------  --------
TOTAL (6)                    0.000014  100.0%      6  0.000002  0.000001  0.000004
```

. . . then `mrkey` will print the following values:

```
command                     output
--------------------------  -------------------------
mrkey profile.txt           FETCH

mrkey --rows=1 profile.txt  FETCH

mrkey -r 1 profile.txt      FETCH

mrkey --rows=2 profile.txt  db file sequential read

mrkey --rows=1-2,6          FETCH
                            db file sequential read
                            PARSE
```

`mrkey` makes it easy to use the output from one `mrskew` report as input into another, such as:

```
$ mrskew V11107_ora_29408.trc --group='$sqlid' --gl=SQLID | tee profile.txt
SQLID                       DURATION       %   CALLS      MEAN       MIN       MAX
--------------------------  --------  ------  ------  --------  --------  --------
8sxr1armhkdun               1.915230   96.0%  11,867  0.000161  0.000000  1.724108
5m2vdrf6kr34q               0.072004    3.6%       4  0.018001  0.000000  0.060003
8fdy1u3wgmf9r               0.008000    0.4%       4  0.002000  0.000000  0.008000
#-512:V11107_ora_29408.trc  0.000000    0.0%       1  0.000000  0.000000  0.000000
--------------------------  --------  ------  ------  --------  --------  --------
TOTAL (4)                   1.995234  100.0%  11,876  0.000168  0.000000  1.724108

$ key=$(mrkey profile.txt)
$ echo "Profile by response time for sqlid '$key'"; \
> mrskew V11107_ora_29408.trc --where="\$sqlid eq '$key'"
Profile by response time for sqlid '8sxr1armhkdun'
CALL-NAME                DURATION       %   CALLS      MEAN       MIN       MAX
-----------------------  --------  ------  ------  --------  --------  --------
FETCH                    1.724108   90.0%       1  1.724108  1.724108  1.724108
```

```
db file sequential read  0.191122   10.0%  11,863  0.000016  0.000006  0.000351
PARSE                    0.000000    0.0%       1  0.000000  0.000000  0.000000
CLOSE                    0.000000    0.0%       1  0.000000  0.000000  0.000000
EXEC                     0.000000    0.0%       1  0.000000  0.000000  0.000000
-----------------------  --------  ------  ------  --------  --------  --------
TOTAL (5)                1.915230  100.0%  11,867  0.000161  0.000000  1.724108
```

Here, the second report explains in detail the time described in the first row of the first report. You could, for example, drill into the second row of the first report by using `key=$(mrkey -r 2 profile.txt)`.

Without `mrkey`, you would have to wait until a report is finished before knowing what `mrskew` command to execute next. With `mrkey`, you can write batch programs that use `mrskew` to drill down into performance data without having to hard-code the key values.

EXIT STATUS

Exit status is 0 on successful completion, and > 0 if an error occurs.

AUTHORS

Cary Millsap, Jeff Holt

SUPPORT

`mrkey` 9.2.1.2

For support, visit https://method-r.com/support.

COPYRIGHT AND LICENSE