NAME

`mrcrop` – Oracle extended SQL trace file cropper

SYNOPSIS

`mrcrop` [`--continued-trace=`*string*] [`--exclude=`*regex*] [`--help` | `-?`] [`--ignore-continued-trace`] [`--license`] [`--man`] [`--ofile=`*string*] [`--recursive` | `-R`] [`--removed-trace=`*string*] [`--silent`] [`--stop-on-exception`] [`--verbose=`*level*] [`--version`] *command* [*args*] [*file* ...]

DESCRIPTION

`mrcrop` extracts the data about a single user experience from one or more Oracle trace files that can contain data about lots of experiences. Each output file is a completely self-contained Oracle trace file, containing file preamble information, all relevant cursor definitions, and all relevant execution plan information for the experience.

It is called `mrcrop` because its atomic operation is reminiscent of cropping unwanted elements out of a picture with a photo editor. You can use `mrcrop` to isolate just one user experience from a set of trace files, or you can use it to loop through all the user experiences it finds, creating a separate, self-contained trace file for each.

`mrcrop` can use any of several algorithms for defining how to partition a file into multiple user experiences (see Commands). Use the *command* argument to specify which algorithm to use.

If *file* is a directory, `mrcrop` will process all files matching the pattern `*.trc` in that directory. The default *file* value is `.`, so running `mrcrop` with no *file* will process files matching `*.trc` in the current working directory. The `--recursive` option defines whether to plunge.

By default, `mrcrop` prints the names of the files it creates. Use the `--verbose` or `--silent` option to control this behavior.

OPTIONS

`--continued-trace=`*oldpath*`::`*newpath* If a `mrcrop` input file `y.trc` contains a line that looks like this,

```
*** TRACE CONTINUED FROM FILE /path/x.trc ***
```

then it is possible that your SQL or PL/SQL statement text will appear in `/path/x.trc` (or some earlier file in a chain of `TRACE CONTINUED FROM`

files) and not `/path/y.trc` itself. `mrcrop` will search backwards through such a chain of trace files to find the appropriate statement text.

However, a problem occurs when your trace files have been moved from where they were created to a different computer for analysis: the files you're looking for no longer reside in `/path`; now they're in some `/newpath`. The `--continued-trace` option lets you instruct `mrcrop` where to look for the files.

For example, imagine that your input file has this line,

```
*** TRACE CONTINUED FROM FILE /u01/trace/PROD_ora_1512.trc ***
```

but `PROD_ora_1512.trc` is found only in `$HOME/traces` on the machine where `mrcrop` executes. The following option would instruct `mrcrop` to look for the file in `$HOME/traces` instead of `/u01/trace`:

```
--continued-trace=/u01/trace::$HOME/traces
```

Each use of `--continued-trace` adds to the list of transformations that `mrcrop` will understand.

`--exclude=`*`regex` Exclude a file from processing if its name matches *regex*. `mrcrop` interprets *regex* as Java Pattern class regular expression syntax. You may use more than one `--exclude` option on a single command invocation. Each use adds to the list of exclusion predicates.

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

`--ignore-continued-trace` Do not process lines that look like those mentioned in `--continued-trace`.

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

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

`--ofile=`*`string`* Use *string* as a format string for the output file name. To illustrate, imagine using the following command:

```
mrcrop island -z=.001 /a/b/c/d_ora_1234.trc
```

`mrcrop` interprets the following symbols in *string*:

Symbol	Subcommand	Description	Example
`%%`	all	'%' character	`%`
`%d`	all	input file directory name	`/a/b/c`
`%b`	all	input file basename	`d_ora_1234`
`%e`	all	input file extension	`.trc`
`%f`	all	input file basename + extension	`d_ora_1234.trc`
`%i`	all	6-digit unique sequence number	`000000`
`%l`	all	input line number	`1492`
`%t`	all	datestamp (yyyyMMddHHmmss)	`20180401121501`

Symbol	Subcommand	Description	Example
%z	island	-z value	1000us

The %t symbol's datestamp is the time at which mrcrop is executed.

The default value is %d/%b/%b-%l%e. The leading %d means that by default, all output files will be created in the same directory as the input file, no matter where you are when you run mrcrop. You can of course override this with your own --ofile option like --ofile=a/b/c/islands/%b-%l%e. If you specify a pathname that does not exist, mrcrop will create it for you.

--recursive, -R If a given *file* is a directory, then process it recursively looking for .trc files. The default is --norecursive.

--removed-trace=*newpath* If a mrcrop input file y.trc contains a line that looks like this,

```
*** TRACE FILE RECREATED AFTER BEING REMOVED ***
```

then it is possible that your SQL or PL/SQL statement text will appear only in that removed file. If you had made a copy of y.trc before you removed it, then you can instruct mrcrop to look for the copy after you restore it to newpath.

For example, imagine that your input file /u01/trace/PROD_ora_4919.trc has this line,

```
*** TRACE FILE RECREATED AFTER BEING REMOVED ***
```

but a copy of the original PROD_ora_4919.trc (before it was removed) is stored as $HOME/trace-backups/PROD_ora_4919.trc on the machine where mrcrop executes. The following option would instruct mrcrop to look for the removed copy of PROD_ora_4919.trc in $HOME/trace-backups.

```
--removed-trace=/u01/trace=$HOME/trace-backups
```

Each use of --removed-trace adds to the list of locations where mrcrop will look for copies of removed files.

--silent An alias for --verbose=0.

--stop-on-exception Stop processing files in the event of an exception in processing one file (a trace file contains no tim values, no datetime stamps, etc.). The default is --nostop-on-exception.

--verbose=*level* If *level* is 1, print to stdout the names of the files created by mrcrop. If *level* is 0, do not print the names. The default is --verbose=1.

--version Print the version number and exit.

COMMANDS

`mrcrop` requires one of the following commands to specify an algorithm for cropping your input:

`mrcrop-datetime(1)` Crop Oracle extended SQL trace files by datetime values.

`mrcrop-experience(1)` Crop Oracle extended SQL trace files by `EXPERIENCE ID` value.

`mrcrop-island(1)` Crop Oracle extended SQL trace files using the oceans–islands–rivers algorithm.

`mrcrop-linerange(1)` Crop Oracle extended SQL trace files by line number range.

`mrcrop-transaction(1)` Crop Oracle extended SQL trace files by `XCTEND` lines.

DISCUSSION

The ideal Oracle extended SQL trace file contains d seconds of execution data for precisely one d-second end-user experience. Such a file is what you need if you want to use `mrprof`. But for a variety of reasons, trace files can contain data that you do not want. For example:

- The business function you want to trace doesn't use Oracle's user session handle attributes (`MODULE`, `ACTION`, or `CLIENTID`), and most of your application users sign in using the same username. So to capture the trace data you want, you activate an `AFTER LOGON` trigger to trace everything executed by that username. Using the trigger for just a few minutes generates thousands of trace files. Many contain no data at all about the experience you're trying to trace, and the trace files that do contain interesting data contain uninteresting data as well. You need to identify the trace data corresponding to the execution of just the one business function you're interested in.
- You have received 67 `.trc` files from an application DBA, representing an uninstrumented connection pool application executing for roughly 100,000 distinct user experiences over the course of about half an hour. Most of these experiences lasted less than a second but ten lasted several seconds. You need to identify the trace data corresponding to these ten experiences that you're interested in.

`mrcrop` helps you crop the background noise out of your trace files so that you can see clearly how each of your interesting application business functions spend your time.

HANDLING LARGE FILE COUNTS

`mrcrop` is built to process huge numbers of files, but working with large file counts can encroach upon the limits of your operating system. For example if the pattern `*.trc` globs into 100,000 file names, your operating system shell may not be able to execute even a basic command like `ls *.trc`. You can avoid the globbing problem with `mrcrop` by giving it a directory name as its input. Even if `d` has 100,000 files in it, using `mrcrop island d` will do what you want without overwhelming your shell.

But there is a limit to how much input data `mrcrop` can handle in a single execution. For example, if you get a `mrcrop: GC overhead limit exceeded` error, then you need to reduce the number of files you process with a single `mrcrop` execution. The `xargs` command makes it easy to stay out of trouble. Executing the following command within the `d` directory will execute `mrcrop island --z=0.1` upon each of its 100,000 files, 256 files per `mrcrop` execution:

```
ls | grep \.trc | xargs -n256 mrcrop island --z=0.1
```

ENVIRONMENT

MRCROP_JVM_OPTS

You can pass options to the Java virtual machine by setting `MRCROP_JVM_OPTS`. For example, you can change `mrcrop`'s maximum Java heap size to 4GB by using `MRCROP_JVM_OPTS=-Xmx4G`. Options set with this environment variable will behave as if they were entered on the command line to the left of the other options that you enter explicitly. You can see all the options you can use with the `java -X` command.

EXIT STATUS

Exit status is 0 for success, 1 for failure.

AUTHORS

Jeff Holt, Cary Millsap

SUPPORT

`mrcrop` 9.2.1.2

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

COPYRIGHT AND LICENSE