NAME

dprofpp - display perl profile data

SYNOPSIS

dprofpp [**-a**|**-z**|**-l**|**-v**|**-U**] [**-d**] [**-s**|**-r**|**-u**] [**-q**] [**-F**] [**-I**|**-E**] [**-O cnt**] [**-A**] [**-R**] [**-S**] [**-g subroutine**] [**-G** <regexp> [**-P**]] [**-f** <regexp>] [profile]

dprofpp **-T** [**-F**] [**-g subroutine**] [profile]

dprofpp **-t** [**-F**] [**-g subroutine**] [profile]

dprofpp **-G** <regexp> [**-P**] [profile]

dprofpp **-p script** [**-Q**] [other opts]

dprofpp **-V** [profile]

DESCRIPTION

The *dprofpp* command interprets profile data produced by a profiler, such as the Devel::DProf profiler. Dprofpp will read the file *tmon.out* and will display the 15 subroutines which are using the most time. By default the times for each subroutine are given exclusive of the times of their child subroutines.

To profile a Perl script run the perl interpreter with the **-d** switch. So to profile script *test.pl* with Devel::DProf the following command should be used.

```
$ perl5 -d:DProf test.pl
```

Then run dprofpp to analyze the profile. The output of dprofpp depends on the flags to the program and the version of Perl you're using.

```
$ dprofpp -u
Total Elapsed Time =    1.67 Seconds
  User Time =    0.61 Seconds
Exclusive Times
%Time Seconds     #Calls sec/call Name
 52.4   0.320          2   0.1600 main::foo
 45.9   0.280        200   0.0014 main::bar
 0.00   0.000          1   0.0000 DynaLoader::import
 0.00   0.000          1   0.0000 main::baz
```

The dprofpp tool can also run the profiler before analyzing the profile data. The above two commands can be executed with one dprofpp command.

```
$ dprofpp -u -p test.pl
```

Consult *"PROFILE FORMAT" in Devel::DProf* for a description of the raw profile.

OUTPUT

Columns are:

%Time

Percentage of time spent in this routine.

#Calls

Number of calls to this routine.

sec/call

Average number of seconds per call to this routine.

Name

Name of routine.

CumulS

Time (in seconds) spent in this routine and routines called from it.

ExclSec

Time (in seconds) spent in this routine (not including those called from it).

Csec/c

Average time (in seconds) spent in each call of this routine (including those called from it).

OPTIONS

-a

Sort alphabetically by subroutine names.

-d

Reverse whatever sort is used

-A

Count timing for autoloaded subroutine as timing for `*::AUTOLOAD`. Otherwise the time to autoload it is counted as time of the subroutine itself (there is no way to separate autoload time from run time).

This is going to be irrelevant with newer Perls. They will inform `Devel::DProf` *when* the `AUTOLOAD` switches to actual subroutine, so a separate statistics for `AUTOLOAD` will be collected no matter whether this option is set.

-R

Count anonymous subroutines defined in the same package separately.

-E

(default) Display all subroutine times exclusive of child subroutine times.

-F

Force the generation of fake exit timestamps if dprofpp reports that the profile is garbled. This is only useful if dprofpp determines that the profile is garbled due to missing exit timestamps. You're on your own if you do this. Consult the BUGS section.

-I

Display all subroutine times inclusive of child subroutine times.

-l

Sort by number of calls to the subroutines. This may help identify candidates for inlining.

-O cnt

Show only *cnt* subroutines. The default is 15.

-p script

Tells dprofpp that it should profile the given script and then interpret its profile data. See **-Q**.

-Q

Used with **-p** to tell dprofpp to quit after profiling the script, without interpreting the data.

-q

Do not display column headers.

-r

Display elapsed real times rather than user+system times.

-s

Display system times rather than user+system times.

-T

Display subroutine call tree to stdout. Subroutine statistics are not displayed.

-t

Display subroutine call tree to stdout. Subroutine statistics are not displayed. When a function is called multiple consecutive times at the same calling level then it is displayed once with a repeat count.

-S

Display *merged* subroutine call tree to stdout. Statistics are displayed for each branch of the tree.

When a function is called multiple (*not necessarily consecutive*) times in the same branch then all these calls go into one branch of the next level. A repeat count is output together with combined inclusive, exclusive and kids time.

Branches are sorted w.r.t. inclusive time.

-U

Do not sort. Display in the order found in the raw profile.

-u

Display user times rather than user+system times.

-V

Print dprofpp's version number and exit. If a raw profile is found then its XS_VERSION variable will be displayed, too.

-v

Sort by average time spent in subroutines during each call. This may help identify candidates for inlining.

-z

(default) Sort by amount of user+system time used. The first few lines should show you which subroutines are using the most time.

-g `subroutine`

Ignore subroutines except `subroutine` and whatever is called from it.

-G <regexp>

Aggregate "Group" all calls matching the pattern together. For example this can be used to group all calls of a set of packages

```
-G "(package1::)|(package2::)|(package3::)"
```

or to group subroutines by name:

```
-G "getNum"
```

-P

Used with -G to aggregate "Pull" together all calls that did not match -G.

-f <regexp>

Filter all calls matching the pattern.

ENVIRONMENT

The environment variable **DPROFPP_OPTS** can be set to a string containing options for dprofpp. You might use this if you prefer **-I** over **-E** or if you want **-F** on all the time.

This was added fairly lazily, so there are some undesirable side effects. Options on the commandline should override options in DPROFPP_OPTS--but don't count on that in this version.

BUGS

Applications which call _exit() or exec() from within a subroutine will leave an incomplete profile. See the **-F** option.

Any bugs in Devel::DProf, or any profiler generating the profile data, could be visible here. See *"BUGS" in Devel::DProf*.

Mail bug reports and feature requests to the perl5-porters mailing list at <*perl5-porters@perl.org*>. Bug reports should include the output of the **-V** option.

FILES

```
dprofpp  - profile processor
tmon.out - raw profile
```

SEE ALSO

perl, *Devel::DProf*, times(2)