

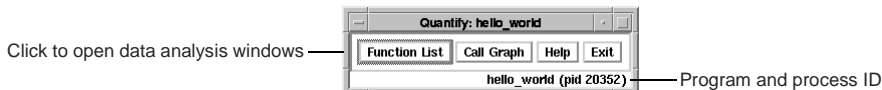
Using Quantify

The CONTROL PANEL appears by default when running a Quantify'd program.

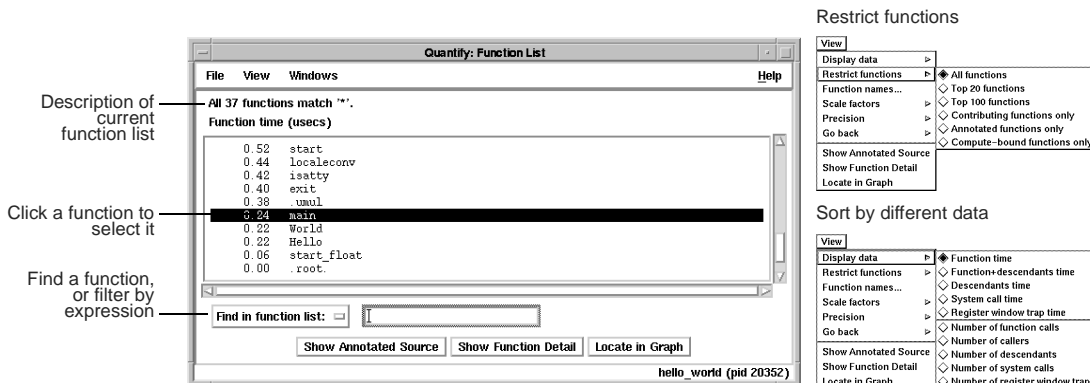
You can also invoke `qv` on a saved `.qv` file:

```
% qv a.out.23.0.qv
```

Control Panel: Open data analysis windows

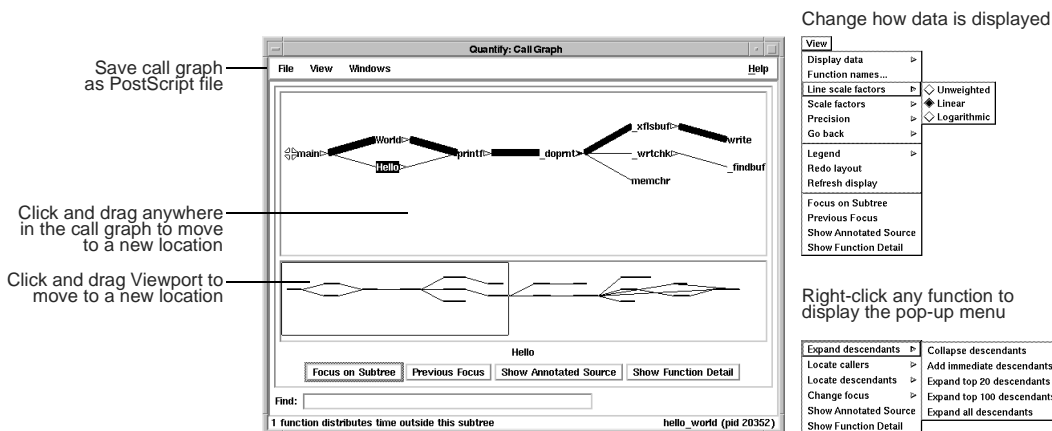


Function List window: Sort functions to find bottlenecks



Call Graph window: Understand your application's calling structure

By default, Quantify expands the top 20 descendents of `.root`.



Rational® Quantify® Quick Reference

Function Detail window: Examine how a function's calling time is distributed

The screenshot shows the 'Quantify: Function Detail' window for the 'malloc' function. The window title is 'Quantify: Function Detail'. The main text area displays the following data:

```

Function name: malloc
Filename: /usr/lib/libc.so.1.9
Called: 1 time
Function time: 96 cycles ( 0.00% of .root.)
Function+descendants time: 30668 cycles ( 0.19% of .root.)
Minimum function time: 96 cycles
Maximum function time: 96 cycles
    
```

Below the main text are two panes: 'Distribution to callers' and 'Contributions from descendants'. The 'Distribution to callers' pane shows:

```

1 time _findbuf
    
```

The 'Contributions from descendants' pane shows:

```

1 time (99.59%) morecore
1 time ( 0.10%) demote
    
```

At the bottom, there is a 'Find:' text box and three buttons: 'Show Annotated Source', 'Show Function Detail', and 'Locate in Graph'. The status bar at the bottom right indicates 'hello_world (pid 20352)'.

Annotations on the left side of the image point to various elements:

- 'Save function detail to file' points to the 'Show Function Detail' button.
- 'All the data collected for the function' points to the main text area.
- 'All the functions that called the selected function' points to the 'Distribution to callers' pane.
- 'Double click a caller or descendant to display its data' points to the entries in the 'Distribution to callers' and 'Contributions from descendants' panes.

Below the screenshot, the text reads: 'All the functions that were called by the selected function'.

Annotated Source window: View line-by-line performance data

The Annotated Source window is available if you compile your program using the `-g` debugging option.

The screenshot shows the 'Quantify: Annotated Source' window for the file 'u25/G2/code/hello_world/hello_world.c'. The window title is 'Quantify: Annotated Source (u25/G2/code/hello_world/hello_world.c)'. The main text area displays the source code with performance annotations:

```

+ Function+descendants time: 63160 cycles ( 0.2425% of .root.)
+ Distribution to Callers:
+ 1 time main
+ .....
99.9953%| { printf("Hello, ");
0.0047%| }

int World()
+ .....
+ Function: World
+ Called: 1 time
+ Function time: 11 cycles ( 0.0000% of .root.)
+ Function+descendants time: 573052 cycles ( 2.2000% of .root.)
+ Distribution to Callers:
+ 1 time main
+ .....
99.9995%| { printf("World.\n");
0.0005%| }

int main()
+ .....
+ Function: main
+ Called: 1 time
    
```

At the bottom, there is a 'Find in source:' text box. The status bar at the bottom right indicates 'hello_world (pid 20352)'.

Annotations on the left side of the image point to various elements:

- 'Save annotated source' points to the 'Show Annotated Source' button.
- 'Function summary' points to the function summary section of the code.
- 'Line-by-line performance annotations' points to the performance data lines in the code.
- 'Find text in source' points to the 'Find in source:' text box.

Conversion characters for filenames

Use these conversion characters when specifying filenames for options.

Character	Converts to	Character	Converts to
%d	Current date (yymmdd)	%t	Current time (hh:mm:ss)
%p	Process id (pid)	%v	Program name
%n	Sequence number, starting at 0. Increments each time the dataset is saved.	%V	Full pathname of the program with "/" replaced by "_"
%T	Thread identifier		

Rational® Quantify® Quick Reference

API functions

Include `<quantifyhome>/quantify.h` in your code and link with `<quantifyhome>/quantify_stubs.a`

Commonly used functions	Description
<code>quantify_help (void)</code>	Prints description of Quantify API functions
<code>quantify_is_running (void)</code>	Returns <code>true</code> if the executable is Quantify'd
<code>quantify_print_recording_state (void)</code>	Prints the recording state of the process
<code>quantify_save_data (void)</code>	Saves data from the start of the program or since last call to <code>quantify_clear_data</code>
<code>quantify_save_data_to_file (char * filename)</code>	Saves data to a file you specify
<code>quantify_add_annotation (char * annotation)</code>	Adds the specified string to the next saved dataset
<code>quantify_clear_data (void)</code>	Clears the performance data recorded to this point
<code>quantify_<action>↑_recording_data (void)</code>	Starts and stops recording of all data
<code>quantify_<action>↑_recording_dynamic_library_data (void)</code>	Starts and stops recording dynamic library data
<code>quantify_<action>↑_recording_register_window_traps (void)</code>	Starts and stops recording register-window trap data
<code>quantify_<action>↑_recording_system_call (char *system_call_string)</code>	Starts and stops recording specific system-call data
<code>quantify_<action>↑_recording_system_calls (void)</code>	Starts and stops recording of all system-call data

† <action> is one of: `start`, `stop`, `is`. For example: `quantify_stop_recording_system_call`

Build-time options

Specify build-time options on the link line to build Quantify'd programs:

```
% quantify -cache-dir=$HOME/cache -always-use-cache-dir cc ...
```

Commonly used build-time options	Default
<code>-always-use-cache-dir</code> Specifies whether Quantify'd files are written to the global cache directory	no
<code>-cache-dir</code> Specifies the global cache directory	<code><quantifyhome>/cache</code>
<code>-collection-granularity</code> Specifies the level of collection granularity	line
<code>-collector</code> Specifies the collect program to handle static constructors in C++ code	not set
<code>-ignore-runtime-environment</code> Prevents the run-time Quantify environment from overriding option values used in building the program	no
<code>-linker</code> Specifies an alternative linker to use instead of the system linker	system-dependent
<code>-use-machine</code> Specifies the build-time analysis of instruction times according to a particular machine	system-dependent

Rational® Quantify® Quick Reference

qv run-time options

To run `qv`, specify the option and the saved `.qv` file: `% qv -write-summary-file a.out.23.qv`

qv options	Default
<code>-add-annotation</code> Specifies a string to add to the binary file	not set
<code>-print-annotations</code> Writes the annotations to <code>stdout</code>	no
<code>-windows</code> Controls whether Quantify runs with the graphical interface	yes
<code>-write-export-file</code> Writes the recorded data in the dataset to a file in export format	not set
<code>-write-summary-file</code> Writes the program summary for the dataset to a file	not set

Run-time options

Specify run-time options using the environment variable `QUANTIFYOPTIONS`:

```
% setenv QUANTIFYOPTIONS "-windows=no"; a.out
```

Commonly used run-time options	Default
<code>-avoid-recording-system-calls</code> Avoids recording specified system calls	system-dependent
<code>-measure-timed-calls</code> Specifies measurement for timing system calls	elapsed-time
<code>-record-child-process-data</code> Records data for child processes created by <code>fork</code> and <code>vfork</code>	no
<code>-record-system-calls</code> Records system calls	yes
<code>-report-excluded-time</code> Reports time that was excluded from the dataset	0.5
<code>-run-at-exit</code> Specifies a shell script to run when the program exits	not set
<code>-run-at-save</code> Specifies a shell script to run each time the program saves counts	not set
<code>-save-data-on-signals</code> Saves data on fatal signals	yes
<code>-save-thread-data</code> Saves composite or per-stack thread data	composite
<code>-write-export-file</code> Writes the dataset to an export file as ASCII text	none
<code>-write-summary-file</code> Writes the program summary for the dataset to a file	<code>/dev/tty</code>
<code>-windows</code> Specifies whether Quantify runs with the graphical interface	yes