GDB Annotations

Edition 0.5
May 1994

Cygnus Support

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Copyright © 1994 Free Software Foundation

1 What is an Annotation?

To produce annotations, start GDB with the `--annotate=2` option.

Annotations start with a newline character, two '`control-z`' characters, and the name of the annotation. If there is no additional information associated with this annotation, the name of the annotation is followed immediately by a newline. If there is additional information, the name of the annotation is followed by a space, the additional information, and a newline. The additional information cannot contain newline characters.

Any output not beginning with a newline and two '`control-z`' characters denotes literal output from GDB. Currently there is no need for GDB to output a newline followed by two '`control-z`' characters, but if there was such a need, the annotations could be extended with an '`escape`' annotation which means those three characters as output.

A simple example of starting up GDB with annotations is:

```
$ gdb --annotate=2
GDB is free software and you are welcome to distribute copies of it
 under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.12.3 (sparc-sun-sunos4.1.3),
Copyright 1994 Free Software Foundation, Inc.

^Z^Zpre-prompt
(gdb)
^Z^Zprompt
quit

^Z^Zpost-prompt
$
```

Here '`quit`' is input to GDB; the rest is output from GDB. The three lines beginning '`^Z^Z`' (where '`^Z`' denotes a '`control-z`' character) are annotations; the rest is output from GDB.

2 The Server Prefix

To issue a command to GDB without affecting certain aspects of the state which is seen by users, prefix it with '`server `'. This means that this command will not affect the command history, nor will it affect GDB's notion of which command to repeat if ⟨RET⟩ is pressed on a line by itself.

The server prefix does not affect the recording of values into the value history; to print a value without recording it into the value history, use the `output` command instead of the `print` command.

3 Values

When a value is printed in various contexts, GDB uses annotations to delimit the value from the surrounding text.

If a value is printed using `print` and added to the value history, the annotation looks like

```
^Z^Zvalue-history-begin history-number value-flags
history-string
^Z^Zvalue-history-value
the-value
^Z^Zvalue-history-end
```

where *history-number* is the number it is getting in the value history, *history-string* is a string, such as '`$5 = `', which introduces the value to the user, *the-value* is the output corresponding to the value itself, and *value-flags* is '`*`' for a value which can be dereferenced and '`-`' for a value which cannot.

If the value is not added to the value history (it is an invalid float or it is printed with the `output` command), the annotation is similar:

```
^Z^Zvalue-begin value-flags
the-value
^Z^Zvalue-end
```

When GDB prints an argument to a function (for example, in the output from the `backtrace` command), it annotates it as follows:

```
^Z^Zarg-begin
argument-name
^Z^Zarg-name-end
separator-string
^Z^Zarg-value value-flags
the-value
^Z^Zarg-end
```

where *argument-name* is the name of the argument, *separator-string* is text which separates the name from the value for the user's benefit (such as '`=`'), and *value-flags* and *the-value* have the same meanings as in a `value-history-begin` annotation.

When printing a structure, GDB annotates it as follows:

```
^Z^Zfield-begin value-flags
field-name
^Z^Zfield-name-end
separator-string
^Z^Zfield-value
the-value
^Z^Zfield-end
```

where *field-name* is the name of the field, *separator-string* is text which separates the name from the value for the user's benefit (such as '`=`'), and *value-flags* and *the-value* have the same meanings as in a `value-history-begin` annotation.

When printing an array, GDB annotates it as follows:

```
^Z^Zarray-section-begin array-index value-flags
```

where *array-index* is the index of the first element being annotated and *value-flags* has the same meaning as in a `value-history-begin` annotation. This is followed by any number of elements, where is element can be either a single element:

```
',' whitespace         ; omitted for the first element
the-value
^Z^Zelt
```

or a repeated element

```
',' whitespace         ; omitted for the first element
the-value
^Z^Zelt-rep number-of-repititions
repetition-string
^Z^Zelt-rep-end
```

In both cases, *the-value* is the output for the value of the element and *whitespace* can contain spaces, tabs, and newlines. In the repeated case, *number-of-repititons* is the number of consecutive array elements which contain that value, and *repetition-string* is a string which is designed to convey to the user that repitition is being depicted.

Once all the array elements have been output, the array annotation is ended with

```
^Z^Zarray-section-end
```

4 Frames

Whenever GDB prints a frame, it annotates it. For example, this applies to frames printed when GDB stops, output from commands such as `backtrace` or `up`, etc.

The frame annotation begins with

```
^Z^Zframe-begin level address
level-string
```

where *level* is the number of the frame (0 is the innermost frame, and other frames have positive numbers), *address* is the address of the code executing in that frame, and *level-string* is a string designed to convey the level to the user. *address* is in the form '`0x`' followed by one or more lowercase hex digits (note that this does not depend on the language). The frame ends with

```
^Z^Zframe-end
```

Between these annotations is the main body of the frame, which can consist of

-
  ```
  ^Z^Zfunction-call
  function-call-string
  ```

  where *function-call-string* is text designed to convey to the user that this frame is associated with a function call made by GDB to a function in the program being debugged.

-

```
^Z^Zsignal-handler-caller
```
signal-handler-caller-string

where *signal-handler-caller-string* is text designed to convey to the user that this frame is associated with whatever mechanism is used by this operating system to call a signal handler (it is the frame which calls the signal handler, not the frame for the signal handler itself).

- A normal frame.

  This can optionally (depending on whether this is thought of as interesting information for the user to see) begin with

  ```
  ^Z^Zframe-address
  ```
  address
  ```
  ^Z^Zframe-address-end
  ```
  separator-string

  where *address* is the address executing in the frame (the same address as in the `frame-begin` annotation, but printed in a form which is intended for user consumption—in particular, the syntax varies depending on the language), and *separator-string* is a string intended to separate this address from what follows for the user's benefit.

  Then comes

  ```
  ^Z^Zframe-function-name
  ```
  function-name
  ```
  ^Z^Zframe-args
  ```
  arguments

  where *function-name* is the name of the function executing in the frame, or '`??`' if not known, and *arguments* are the arguments to the frame, with parentheses around them (each argument is annotated individually as well see Chapter 3 [Values], page 2).

  If source information is available, a reference to it is then printed:

  ```
  ^Z^Zframe-source-begin
  ```
  source-intro-string
  ```
  ^Z^Zframe-source-file
  ```
  filename
  ```
  ^Z^Zframe-source-file-end
  :
  ^Z^Zframe-source-line
  ```
  line-number
  ```
  ^Z^Zframe-source-end
  ```

  where *source-intro-string* separates for the user's benefit the reference from the text which precedes it, *filename* is the name of the source file, and *line-number* is the line number within that file (the first line is line 1).

  If GDB prints some information about where the frame is from (which library, which load segment, etc.; currently only done on the RS/6000), it is annotated with

  ```
  ^Z^Zframe-where
  ```
  information

  Then, if source is to actually be displayed for this frame (for example, this is not true for output from the `backtrace` command), then a `source` annotation (see Chapter 11

[Source], page 8) is displayed. Unlike most annotations, this is output instead of the normal text which would be output, not in addition.

5 Displays

When GDB is told to display something using the `display` command, the results of the display are annotated:

```
^Z^Zdisplay-begin
number
^Z^Zdisplay-number-end
number-separator
^Z^Zdisplay-format
format
^Z^Zdisplay-expression
expression
^Z^Zdisplay-expression-end
expression-separator
^Z^Zdisplay-value
value
^Z^Zdisplay-end
```

where *number* is the number of the display, *number-separator* is intended to separate the number from what follows for the user, *format* includes information such as the size, format, or other information about how the value is being displayed, *expression* is the expression being displayed, *expression-separator* is intended to separate the expression from the text that follows for the user, and *value* is the actual value being displayed.

6 Annotation for GDB Input

When GDB prompts for input, it annotates this fact so it is possible to know when to send output, when the output from a given command is over, etc.

Different kinds of input each have a different *input type*. Each input type has three annotations: a `pre-` annotation, which denotes the beginning of any prompt which is being output, a plain annotation, which denotes the end of the prompt, and then a `post-` annotation which denotes the end of any echo which may (or may not) be associated with the input. For example, the `prompt` input type features the following annotations:

```
^Z^Zpre-prompt
^Z^Zprompt
^Z^Zpost-prompt
```

The input types are

`prompt` When GDB is prompting for a command (the main GDB prompt).

`commands` When GDB prompts for a set of commands, like in the `commands` command. The annotations are repeated for each command which is input.

`overload-choice`
When GDB wants the user to select between various overloaded functions.

`query` When GDB wants the user to confirm a potentially dangerous operation.

`prompt-for-continue`
: When GDB is asking the user to press return to continue. Note: Don't expect this to work well; instead use `set height 0` to disable prompting. This is because the counting of lines is buggy in the presence of annotations.

7 Errors

```
^Z^Zquit
```

This annotation occurs right before GDB responds to an interrupt.

```
^Z^Zerror
```

This annotation occurs right before GDB responds to an error.

Quit and error annotations indicate that any annotations which GDB was in the middle of may end abruptly. For example, if a `value-history-begin` annotation is followed by a `error`, one cannot expect to receive the matching `value-history-end`. One cannot expect not to receive it either, however; an error annotation does not necessarily mean that GDB is immediately returning all the way to the top level.

A quit or error annotation may be preceded by

```
^Z^Zerror-begin
```

Any output between that and the quit or error annotation is the error message.

Warning messages are not yet annotated.

8 Information on Breakpoints

The output from the `info breakpoints` command is annotated as follows:

```
^Z^Zbreakpoints-headers
header-entry
^Z^Zbreakpoints-table
```

where *header-entry* has the same syntax as an entry (see below) but instead of containing data, it contains strings which are intended to convey the meaning of each field to the user. This is followed by any number of entries. If a field does not apply for this entry, it is omitted. Fields may contain trailing whitespace. Each entry consists of:

```
^Z^Zrecord
^Z^Zfield 0
number
^Z^Zfield 1
type
^Z^Zfield 2
disposition
^Z^Zfield 3
enable
^Z^Zfield 4
address
```

```
^Z^Zfield 5
```
what
```
^Z^Zfield 6
```
frame
```
^Z^Zfield 7
```
condition
```
^Z^Zfield 8
```
ignore-count
```
^Z^Zfield 9
```
commands

Note that *address* is intended for user consumption—the syntax varies depending on the language.

The output ends with

```
^Z^Zbreakpoints-table-end
```

9 Invalidation Notices

The following annotations say that certain pieces of state may have changed.

`^Z^Zframes-invalid`
: The frames (for example, output from the `backtrace` command) may have changed.

`^Z^Zbreakpoints-invalid`
: The breakpoints may have changed. For example, the user just added or deleted a breakpoint.

10 Running the Program

When the program starts executing due to a GDB command such as `step` or `continue`,

```
^Z^Zstarting
```

is output. When the program stops,

```
^Z^Zstopped
```

is output. Before the `stopped` annotation, a variety of annotations describe how the program stopped.

`^Z^Zexited` *exit-status*
: The program exited, and *exit-status* is the exit status (zero for successful exit, otherwise nonzero).

`^Z^Zsignalled`
: The program exited with a signal. After the `^Z^Zsignalled`, the annotation continues:

> *intro-text*
> `^Z^Zsignal-name`
> *name*
> `^Z^Zsignal-name-end`
> *middle-text*
> `^Z^Zsignal-string`
> *string*
> `^Z^Zsignal-string-end`
> *end-text*

where *name* is the name of the signal, such as `SIGILL` or `SIGSEGV`, and *string* is the explanation of the signal, such as `Illegal Instruction` or `Segmentation fault`. *intro-text*, *middle-text*, and *end-text* are for the user's benefit and have no particular format.

`^Z^Zsignal`
: The syntax of this annotation is just like `signalled`, but GDB is just saying that the program received the signal, not that it was terminated with it.

`^Z^Zbreakpoint` *number*
: The program hit breakpoint number *number*.

`^Z^Zwatchpoint` *number*
: The program hit watchpoint number *number*.

11 Displaying Source

The following annotation is used instead of displaying source code:

`^Z^Zsource` *filename*`:`*line*`:`*character*`:`*middle*`:`*addr*

where *filename* is an absolute file name indicating which source file, *line* is the line number within that file (where 1 is the first line in the file), *character* is the character position within the file (where 0 is the first character in the file) (for most debug formats this will necessarily point to the beginning of a line), *middle* is '`middle`' if *addr* is in the middle of the line, or '`beg`' if *addr* is at the beginning of the line, and *addr* is the address in the target program associated with the source which is being displayed. *addr* is in the form '`0x`' followed by one or more lowercase hex digits (note that this does not depend on the language).

12 Annotations We Might Want in the Future

```
- target-invalid
  the target might have changed (registers, heap contents, or
  execution status).  For performance, we might eventually want
  to hit 'registers-invalid' and 'all-registers-invalid' with
  greater precision

- systematic annotation for set/show parameters (including
  invalidation notices).
```

- similarly, ‘info’ returns a list of candidates for invalidation notices.

Index

(Index is nonexistent)