Btparser

A program failure analysis library

Karel Klíč

August 27, 2012

Contents

Chapter 1

Overview

Part I

Concepts

Part II

Implementation

Chapter 2

Overview

Btparser is implemented in the C language as defined in the C99 standard (ISO/IEC 9899:1999). It uses the C standard library as well as some additional libraries:

- elfutils

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

3.2 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 btp_callgraph Struct Reference

Collaboration diagram for btp_callgraph:

Data Fields

- uint64_t **address**
- uint64_t * **callees**
- struct btp_callgraph * **next**

The documentation for this struct was generated from the following file:

- callgraph.h

4.2 btp_cluster Struct Reference

`#include <cluster.h>`Collaboration diagram for btp_cluster:

Data Fields

- int **size**
- int * **objects**
- struct btp_cluster * **next**

4.2.1 Detailed Description

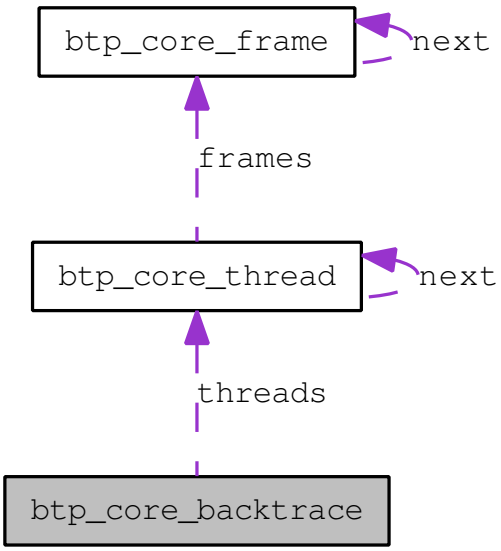
Represents a cluster of objects.

The documentation for this struct was generated from the following file:

- cluster.h

4.3 btp_core_backtrace Struct Reference

Collaboration diagram for btp_core_backtrace:



Data Fields

- enum btp_core_backtrace_type **type**
- struct btp_core_thread * **threads**

The documentation for this struct was generated from the following file:

- core_backtrace.h

4.4 btp_core_frame Struct Reference

`#include <core_frame.h>`Collaboration diagram for btp_core_frame:

Data Fields

- uint64_t address
- char * build_id
- uint64_t **build_id_offset**
- char * **function_name**
- char * **file_name**
- char * fingerprint
- struct btp_core_frame * next

4.4.1 Detailed Description

A frame representing a function call on a call stack of a thread.

4.4.2 Field Documentation

4.4.2.1 uint64_t btp_core_frame::address

Address of the machine code in memory. This is useful only when build_id is not present for some reason. For example, this might be a null dereference (address is 0) or calling a method from null class pointer (address is a low number -- offset to the class).

Some programs generate machine code during runtime (JavaScript engines, JVM, the Gallium llvmpipe driver).

4.4.2.2 char* btp_core_frame::build_id

Build id of the ELF binary. It might be NULL if the frame does not point to memory with code.

4.4.2.3 char* btp_core_frame::fingerprint

Hash of the function contents.

4.4.2.4 struct btp_core_frame* btp_core_frame::next [read]

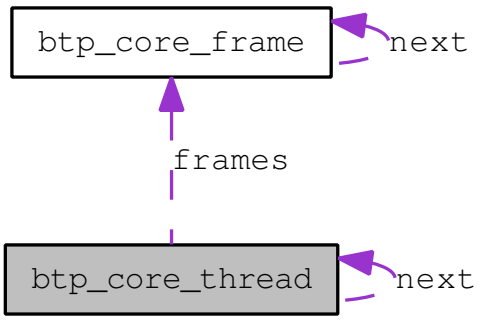
A sibling frame residing below this one, or NULL if this is the last frame in the parent thread.

The documentation for this struct was generated from the following file:

- core_frame.h

4.5 btp_core_thread Struct Reference

`#include <core_thread.h>`Collaboration diagram for btp_core_thread:



Data Fields

- struct btp_core_frame * frames
- struct btp_core_thread * next

4.5.1 Detailed Description

Represents a thread containing frames.

4.5.2 Field Documentation

4.5.2.1 struct btp_core_frame* btp_core_thread::frames [read]

Thread's frames, starting from the top of the stack.

4.5.2.2 struct btp_core_thread* btp_core_thread::next [read]

A sibling thread, or NULL if this is the last thread in a backtrace.

The documentation for this struct was generated from the following file:

- core_thread.h

4.6 btp_dendrogram Struct Reference

`#include <cluster.h>`

Data Fields

- int **size**
- int * **order**
- float * merge_levels

4.6.1 Detailed Description

Represents a dendrogram created by clustering.

4.6.2 Field Documentation

4.6.2.1 float* btp_dendrogram::merge_levels

Levels at which the clusters were merged. The clustering can be reconstructed in order of increasing levels. There are (size - 1) levels.

The documentation for this struct was generated from the following file:

- cluster.h

4.7 btp_disasm_state Struct Reference

Data Fields

- bfd ∗ **bfd_file**
- disassembler_ftype **disassembler**
- struct disassemble_info **info**
- char ∗ **error_message**

The documentation for this struct was generated from the following file:

- disassembler.h

4.8 btp_distances Struct Reference

`#include <metrics.h>`

Data Fields

- int **m**
- int **n**
- float * **distances**

4.8.1 Detailed Description

Represents an m-by-n distance matrix. (only entries (i, j) where i < j are actually stored)

The documentation for this struct was generated from the following file:

- metrics.h

4.9 btp_elf_frame_description_entry Struct Reference

`#include <elves.h>`Collaboration diagram for btp_elf_frame_description_entry:

Data Fields

- uint64_t start_address
- uint64_t length
- struct btp_elf_frame_description_entry * **next**

4.9.1 Detailed Description

A Frame Description Entry (FDE) representing items in the .eh_frame section in ELF binaries.

4.9.2 Field Documentation

4.9.2.1 uint64_t btp_elf_frame_description_entry::length

Length of the function in bytes.

4.9.2.2 uint64_t btp_elf_frame_description_entry::start_address

Offset where a function starts. If the function is present in the Procedure Linkage Table, this address matches some address in btp_elf_plt_entry.

The documentation for this struct was generated from the following file:

- elves.h

4.10 btp_elf_plt_entry Struct Reference

`#include <elves.h>`Collaboration diagram for btp_elf_plt_entry:

Data Fields

- uint64_t address
- char * symbol_name
- struct btp_elf_plt_entry * **next**

4.10.1 Detailed Description

File name elf.h cannot be used due to collision with <elf.h> system include. An entry of the Procedure Linkage Table (PLT).

4.10.2 Field Documentation

4.10.2.1 uint64_t btp_elf_plt_entry::address

Address of the entry.

4.10.2.2 char* btp_elf_plt_entry::symbol_name

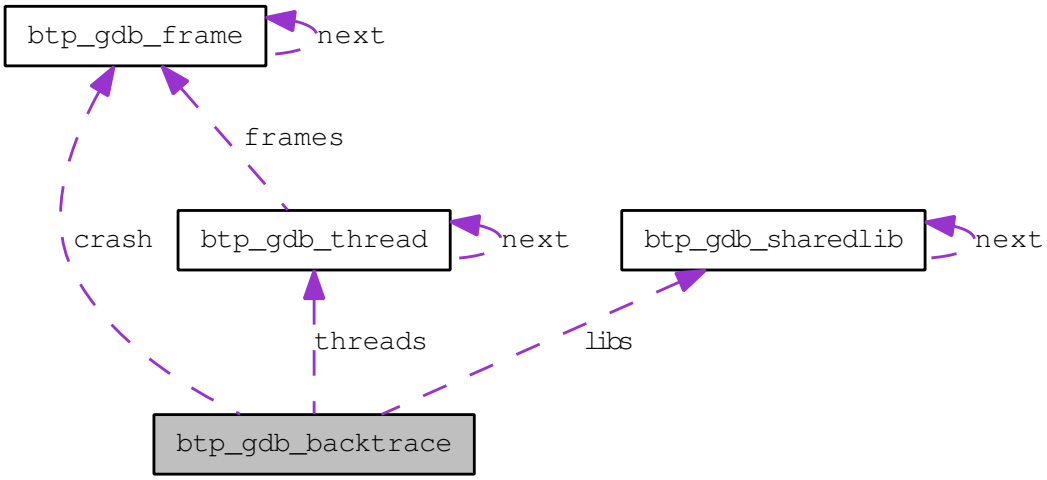
Symbol name corresponding to the address.

The documentation for this struct was generated from the following file:

- elves.h

4.11 btp_gdb_backtrace Struct Reference

`#include <gdb_backtrace.h>`Collaboration diagram for btp_gdb_backtrace:



Data Fields

- struct btp_gdb_thread ∗ **threads**
- struct btp_gdb_frame ∗ crash
- struct btp_gdb_sharedlib ∗ libs

4.11.1 Detailed Description

A backtrace obtained at the time of a program crash, consisting of several threads which contains frames.

This structure represents a backtrace as produced by the GNU Debugger.

4.11.2 Field Documentation

4.11.2.1 struct btp_gdb_frame∗ btp_gdb_backtrace::crash [read]

The frame where the crash happened according to debugger. It might be that we can not tell to which thread this frame belongs, because some threads end with mutually indistinguishable frames.

4.11.2.2 struct btp_gdb_sharedlib∗ btp_gdb_backtrace::libs [read]

Shared libraries loaded at the moment of crash.

The documentation for this struct was generated from the following file:

- gdb_backtrace.h

4.12 btp_gdb_frame Struct Reference

`#include <gdb_frame.h>`Collaboration diagram for btp_gdb_frame:

Data Fields

- char * function_name
- char * function_type
- unsigned number
- char * source_file
- unsigned source_line
- bool signal_handler_called
- uint64_t address
- char * library_name
- struct btp_gdb_frame * next

4.12.1 Detailed Description

A frame representing a function call or a signal handler on a call stack of a thread.

4.12.2 Field Documentation

4.12.2.1 uint64_t btp_gdb_frame::address

The function address in the computer memory, or -1 when the address is unknown. Address is unknown when the frame represents inlined function.

4.12.2.2 char* btp_gdb_frame::function_name

A function name or NULL. If it's NULL, signal_handler_called is true.

4.12.2.3 char* btp_gdb_frame::function_type

A function type, or NULL if it isn't present.

4.12.2.4 char* btp_gdb_frame::library_name

A library name or NULL.

4.12.2.5 struct btp_gdb_frame* btp_gdb_frame::next [read]

A sibling frame residing below this one, or NULL if this is the last frame in the parent thread.

4.12.2.6 unsigned btp_gdb_frame::number

A frame number in a thread. It does not necessarily show the actual position in the thread, as this number is set by the parser and never updated.

4.12.2.7 bool btp_gdb_frame::signal_handler_called

Signal handler was called on this frame.

4.12.2.8 char* btp_gdb_frame::source_file

The name of the source file containing the function definition, or the name of the binary file (.so) with the binary code of the function, or NULL.

4.12.2.9 unsigned btp_gdb_frame::source_line

A line number in the source file, determining the position of the function definition, or -1 when unknown.

The documentation for this struct was generated from the following file:

- gdb_frame.h

4.13 btp_gdb_sharedlib Struct Reference

Collaboration diagram for btp_gdb_sharedlib:

Data Fields

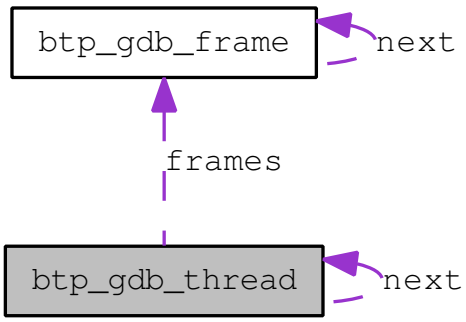
- uint64_t **from**
- uint64_t **to**
- int **symbols**
- char * **soname**
- struct btp_gdb_sharedlib * **next**

The documentation for this struct was generated from the following file:

- gdb_sharedlib.h

4.14 btp_gdb_thread Struct Reference

`#include <gdb_thread.h>`Collaboration diagram for btp_gdb_thread:



Data Fields

- unsigned **number**
- struct btp_gdb_frame ∗ frames
- struct btp_gdb_thread ∗ next

4.14.1 Detailed Description

Represents a thread containing frames.

4.14.2 Field Documentation

4.14.2.1 struct btp_gdb_frame∗ btp_gdb_thread::frames [read]

Thread's frames, starting from the top of the stack.

4.14.2.2 struct btp_gdb_thread∗ btp_gdb_thread::next [read]

A sibling thread, or NULL if this is the last thread in a backtrace.

The documentation for this struct was generated from the following file:

- gdb_thread.h

4.15 btp_location Struct Reference

`#include <location.h>`

Data Fields

- int line
- int column
- const char * message

4.15.1 Detailed Description

A location in the backtrace file with an attached message. It's used for error reporting: the line and the column points to the place where a parser error occurred, and the message explains what the parser expected and didn't find on that place.

4.15.2 Field Documentation

4.15.2.1 int btp_location::column

Starts from 0.

4.15.2.2 int btp_location::line

Starts from 1.

4.15.2.3 const char* btp_location::message

Error message related to the line and column. Do not release the memory this pointer points to.

The documentation for this struct was generated from the following file:

- location.h

4.16 btp_sha1_ctx Struct Reference

Data Fields

- uint8_t **wbuffer** [64]
- uint64_t **total64**
- uint32_t **hash** [8]

The documentation for this struct was generated from the following file:

- sha1.h

4.17 btp_strbuf Struct Reference

Data Fields

- int alloc
- int len
- char * **buf**

4.17.1 Field Documentation

4.17.1.1 int btp_strbuf::alloc

Size of the allocated buffer. Always > 0.

4.17.1.2 int btp_strbuf::len

Length of the string, without the ending .

The documentation for this struct was generated from the following file:

- strbuf.h

4.18 btp_unstrip_entry Struct Reference

`#include <unstrip.h>`Collaboration diagram for btp_unstrip_entry:

Data Fields

- uint64_t **start**
- uint64_t **length**
- char * **build_id**
- char * **file_name**
- char * **mod_name**
- struct btp_unstrip_entry * **next**

4.18.1 Detailed Description

Output of the unstrip utility.

The documentation for this struct was generated from the following file:

- unstrip.h

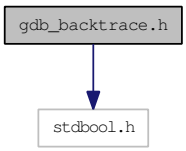
Chapter 5

File Documentation

5.1 gdb_backtrace.h File Reference

`#include <stdbool.h>`

Include dependency graph for gdb_backtrace.h:



Data Structures

- struct btp_gdb_backtrace

Functions

- struct btp_gdb_backtrace * btp_gdb_backtrace_new ()
- void btp_gdb_backtrace_init (struct btp_gdb_backtrace *backtrace)
- void btp_gdb_backtrace_free (struct btp_gdb_backtrace *backtrace)
- struct btp_gdb_backtrace * btp_gdb_backtrace_dup (struct btp_gdb_backtrace *backtrace)
- int btp_gdb_backtrace_get_thread_count (struct btp_gdb_backtrace *backtrace)
- void btp_gdb_backtrace_remove_threads_except_one (struct btp_gdb_backtrace *backtrace, struct btp_gdb_thread *thread)
- struct btp_gdb_thread * btp_gdb_backtrace_find_crash_thread (struct btp_gdb_backtrace *backtrace)
- void btp_gdb_backtrace_limit_frame_depth (struct btp_gdb_backtrace *backtrace, int depth)
- float btp_gdb_backtrace_quality_simple (struct btp_gdb_backtrace *backtrace)
- float btp_gdb_backtrace_quality_complex (struct btp_gdb_backtrace *backtrace)
- char * btp_gdb_backtrace_to_text (struct btp_gdb_backtrace *backtrace, bool verbose)
- struct btp_gdb_frame * btp_gdb_backtrace_get_crash_frame (struct btp_gdb_backtrace *backtrace)
- char * btp_backtrace_get_duplication_hash (struct btp_gdb_backtrace *backtrace)

- struct btp_gdb_backtrace * btp_gdb_backtrace_parse (const char **input, struct btp_location *location)
- bool btp_gdb_backtrace_parse_header (const char **input, struct btp_gdb_frame **frame, struct btp_location *location)
- void btp_gdb_backtrace_set_libnames (struct btp_gdb_backtrace *backtrace)
- struct btp_gdb_thread * btp_gdb_backtrace_get_optimized_thread (struct btp_gdb_backtrace *backtrace, int max_frames)

5.1.1 Detailed Description

Backtrace as produced by GDB (the GNU Project Debugger).

5.1.2 Function Documentation

5.1.2.1 char* btp_backtrace_get_duplication_hash (struct btp_gdb_backtrace * *backtrace*)

Calculates the duplication hash string of the backtrace.

Parameters:

backtrace It must be non-NULL pointer. It's not modified by calling this function.

Returns:

This function never returns NULL. The caller is responsible for releasing the returned memory using function free().

5.1.2.2 struct btp_gdb_backtrace* btp_gdb_backtrace_dup (struct btp_gdb_backtrace * *backtrace*) [read]

Creates a duplicate of the backtrace.

Parameters:

backtrace The backtrace to be copied. It's not modified by this function.

Returns:

This function never returns NULL. The returned duplicate must be released by calling the function btp_gdb_backtrace_free().

5.1.2.3 struct btp_gdb_thread* btp_gdb_backtrace_find_crash_thread (struct btp_gdb_backtrace * *backtrace*) [read]

Searches all threads and tries to find the one that caused the crash. It might return NULL if the thread cannot be determined.

Parameters:

backtrace It must be non-NULL pointer. It's not modified by calling this function.

5.1.2.4 void btp_gdb_backtrace_free (struct btp_gdb_backtrace * *backtrace*)

Releases the memory held by the backtrace, its threads, frames, shared libraries.

Parameters:

backtrace If the backtrace is NULL, no operation is performed.

5.1.2.5 struct btp_gdb_frame* btp_gdb_backtrace_get_crash_frame (struct btp_gdb_backtrace * *backtrace*) [read]

Analyzes the backtrace to get the frame where a crash occurred.

Parameters:

backtrace It must be non-NULL pointer. It's not modified by calling this function.

Returns:

The returned value must be released by calling btp_gdb_frame_free() when it's no longer needed, because it is a deep copy of the crash frame from the backtrace. NULL is returned if the crash frame is not found.

5.1.2.6 struct btp_gdb_thread* btp_gdb_backtrace_get_optimized_thread (struct btp_gdb_backtrace * *backtrace*, int *max_frames*) [read]

Return crash thread optimized for comparison. It's normalized, with library names set and functions without names (signal handlers) are removed.

Parameters:

backtrace It must be non-NULL pointer. It's not modified by calling this function.

max_frames The maximum number of frames in the returned crash thread. Superfluous frames are removed from the returned thread.

Returns:

A newly allocated thread structure or NULL. NULL is returned when the crashing thread could not be found. The returned structure should be released by btp_gdb_thread_free() by the caller.

5.1.2.7 int btp_gdb_backtrace_get_thread_count (struct btp_gdb_backtrace * *backtrace*)

Returns a number of threads in the backtrace.

Parameters:

backtrace It's not modified by calling this function.

5.1.2.8 void btp_gdb_backtrace_init (struct btp_gdb_backtrace * *backtrace*)

Initializes all members of the backtrace structure to their default values. No memory is released, members are simply overwritten. This is useful for initializing a backtrace structure placed on the stack.

5.1.2.9 void btp_gdb_backtrace_limit_frame_depth (struct btp_gdb_backtrace * *backtrace*, int *depth*)

Remove frames from the bottom of threads in the backtrace, until all threads have at most 'depth' frames.

Parameters:

backtrace Must be non-NULL pointer.

5.1.2.10 struct btp_gdb_backtrace* btp_gdb_backtrace_new () [read]

Creates and initializes a new backtrace structure.

Returns:

It never returns NULL. The returned pointer must be released by calling the function btp_gdb_-backtrace_free().

5.1.2.11 struct btp_gdb_backtrace* btp_gdb_backtrace_parse (const char ** *input*, struct btp_location * *location*) [read]

Parses a textual backtrace and puts it into a structure. If parsing fails, the input parameter is not changed and NULL is returned.

```
struct btp_location location;
btp_location_init(&location);
char *input = "...";
struct btp_gdb_backtrace *backtrace;
backtrace = btp_gdb_backtrace_parse(input, location);
if (!backtrace)
{
  fprintf(stderr,
          "Failed to parse the backtrace.\n"
          "Line %d, column %d: %s\n",
          location.line,
          location.column,
          location.message);
  exit(-1);
}
btp_gdb_backtrace_free(backtrace);
```

Parameters:

input Pointer to the string with the backtrace. If this function returns true, this pointer is modified to point after the backtrace that was just parsed.

location The caller must provide a pointer to an instance of btp_location here. The line and column members of the location are gradually increased as the parser handles the input, so the location should be initialized by btp_location_init() before calling this function to get reasonable values. When this function returns false (an error occurred), the structure will contain the error line, column, and message.

Returns:

A newly allocated backtrace structure or NULL. A backtrace struct is returned when at least one thread was parsed from the input and no error occurred. The returned structure should be released by btp_-gdb_backtrace_free().

5.1.2.12 bool btp_gdb_backtrace_parse_header (const char ** *input*, struct btp_gdb_frame ** *frame*, struct btp_location * *location*)

Parse backtrace header if it is available in the backtrace. The header usually contains frame where the program crashed.

Parameters:

input Pointer that will be moved to point behind the header if the header is successfully detected and parsed.

frame If this function succeeds and returns true, *frame contains the crash frame that is usually a part of the header. If no frame is detected in the header, *frame is set to NULL.

```
[New Thread 11919]
[New Thread 11917]
Core was generated by ‘evince file:
Program terminated with signal 8, Arithmetic exception.
#0  0x000000322a2362b9 in repeat (image=<value optimized out>,
    mask=<value optimized out>, mask_bits=<value optimized out>)
    at pixman-bits-image.c:145
145     pixman-bits-image.c: No such file or directory.
        in pixman-bits-image.c
```

5.1.2.13 float btp_gdb_backtrace_quality_complex (struct btp_gdb_backtrace * *backtrace*)

Evaluates the quality of the backtrace. The quality is determined depending on the ratio of frames with function name fully known to all frames.

Parameters:

backtrace It must be non-NULL pointer. It's not modified by calling this function.

Returns:

A number between 0 and 1. 0 means the lowest quality, 1 means full backtrace is known. The returned value takes into account that the thread which caused the crash is more important than the other threads, and the frames around the crash frame are more important than distant frames.

5.1.2.14 float btp_gdb_backtrace_quality_simple (struct btp_gdb_backtrace * *backtrace*)

Evaluates the quality of the backtrace. The quality is the ratio of the number of frames with function name fully known to the number of all frames. This function does not take into account that some frames are more important than others.

Parameters:

backtrace It must be non-NULL pointer. It's not modified by calling this function.

Returns:

A number between 0 and 1. 0 means the lowest quality, 1 means full backtrace is known (all function names are known).

5.1.2.15 void btp_gdb_backtrace_remove_threads_except_one (struct btp_gdb_backtrace * *backtrace*, struct btp_gdb_thread * *thread*)

Removes all threads from the backtrace and deletes them, except the one provided as a parameter.

Parameters:

thread This function does not check whether the thread is a member of the backtrace. If it's not, all threads are removed from the backtrace and then deleted.

5.1.2.16 void btp_gdb_backtrace_set_libnames (struct btp_gdb_backtrace * *backtrace*)

Set library names in all frames in the backtrace according to the the sharedlib data.

5.1.2.17 char* btp_gdb_backtrace_to_text (struct btp_gdb_backtrace * *backtrace*, bool *verbose*)

Returns textual representation of the backtrace.

Parameters:

backtrace It must be non-NULL pointer. It's not modified by calling this function.

Returns:

This function never returns NULL. The caller is responsible for releasing the returned memory using function free().

Chapter 6

Known Bugs

Empty.

Index