tracefct

keeping track of where you've been - a function tracer
edition 1.0.10 for `tracefct` version 1.0.10
29 July 2011

Diab Jerius
Dan Nguyen

Table of Contents

1 Copying

2 Introduction

`tracefct` is a package of routines which help a program keep track of its function calling sequence so that it can provide that information on fatal errors. It also provides a uniform format for messages to `stderr` which identify the process producing the messages.

Messages have the following appearance:

```
# pid: exec-name: message
```

where *pid* is the process id number, *exec-name* is the name of the executable, and *message* is a user provided message.

2.1 Usage

The `tracefct` library is initialized with `tf_init()`. This should be called before any other routine, as it registers the name of the executable. Directly after entering a function, `tf_enter()` should be called with the name of the function. Before leaving a function `tf_leave()` should be called. If a fatal error occurs, `tf_exit()` should be called with a return value that will be passed to the invoking process and a message to be written to `stderr`. Interesting return values are predefined in the '`exiterrvals.h`' header file. `tf_exit()` does *not* return to the calling function, but calls `exit()`. Warning or diagnostic messages may be written via `tf_message` or `tf_vmessage`. The function stack may be written at any time with `tf_dump_stack()`.

`tracefct` has a predefined output stream of `stderr`. However, it is sometimes useful to have the output sent to another file instead. This is accomplished with the `tf_open` function. In the case that the `tracefct` is not to `stderr`, the standard prefix is *not* printed. The output stream may be reset to `stderr` with the `tf_close` function. Note that the `tf_exit` will write both to `stderr` and the `tracefct` output stream, provided that they are different.

3 Library Routines

3.1 tf_assert

program verification

Synopsis

```
#include <tracefct.h>

tf_assert(expression);
tf_assert1(expression);
tf_assert2(expression);
tf_assert3(expression);
tf_assert4(expression);
tf_assert5(expression);
```

Parameters

`expression`
: A standard C expression.

Description

The `tf_assert` routines are C preprocessor macros that indicate that *expression* is expected to be true at the point in the program that the macros are invoked. If *expression* is false, a diagnostic message is printed to `stderr` indicating the file and line number at which the assertion failed, and the program exits.

Unlike the standard `assert` macro, `tf_assert` is *always* valid. The other routines are valid depending upon the value of the preprocessor macro `TF_ASSERT_LEVEL`. The macro `tf_assert`*n* is valid if '`TF_ASSERT_LEVEL >= `*n*'. By default, `TF_ASSERT_LEVEL` is undefined (equivalently set to '`0`'). This setup provides a *compile* time choice of diagnostic output.

The benefit of these routines over the standard `assert` macro is that they use the `tracefct` standard output format.

3.2 tf_init

Initialize the function trace stack.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_init(
  const char *name,
  int print_it,
  int num_fct_to_print
```

```
);
```

Parameters

`const char *name`
: the name of the program, preferrably argv[0]. path information (if present) is not stripped.

`int print_it`
: boolean flag. if true, a message will be output upon entry and exit of a function.

`int num_fct_to_print`
: if positive, only the requested number of function names will be printed. if non-positive, all of the function names will be printed.

Description

This routine initializes the function stack as well as registering the name of the executable. The calling routine can also set up some output options which determine the depth of the function stack to print and whether a diagnostic should be printed at every entry and return of a function. `tracefct` stores the passed pointer to the program name, it does *not* make a copy of the string. The calling function mustn't invalidate the pointer.

3.3 tf_enter

Register function.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_enter(const char *name);
```

Parameters

`const char *name`
: the name of the function.

Description

This routine is called upon entry into a function. It registers the function's name so that it can be used in messages. Note that it saves the pointer to the function name, it does *not* make a copy of the string.

3.4 tf_leave

Deregister a function.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_leave(void);
```

Description

This routine is called upon exit from a function. It deregisters the last registered function.

3.5 tf_die

Exit a program, dumping the function stack.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_die(
  const char *format,
  ...
);
```

Parameters

`const char *format`
: a printf style format string. passed to vsprintf

`...`
: additional arguments to be passed to vfprintf

Description

This function prints an error message to `stderr` as well as the `tracefct` output stream (if different from `stderr`) and then exits the program with a value of `EXIT_FAILURE`. The error message is passed in the same fashion as the arguments to `vprintf` or `sprintf`. The message may contain multiple output lines (i.e., multiple newline characters). If a trailing newline character is not specified, it will be appended.

Contrast this with `tf_exit`.

3.6 tf_exit

Exit a program, dumping the function stack.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_exit(
  int exit_code,
```

```
  const char *format,
  ...
);
```

Parameters

`int exit_code`
: the exit code to be returned to the system

`const char *format`
: a printf style format string. passed to vsprintf

`...`
: additional arguments to be passed to vfprintf

Description

This function prints an error message to `stderr` as well as the `tracefct` output stream (if different from `stderr`) and then exits the program with the supplied error code. Interesting error codes are predefined in '`exiterrvals.h`'. The error message is passed in the same fashion as the arguments to `vprintf` or `sprintf`. The message may contain multiple output lines (i.e., multiple newline characters). If a trailing newline character is not specified, it will be appended.

3.7 tf_message

Print a formatted message to the `tracefct` output stream.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_message(
  const char *format,
  ...
);
```

Parameters

`const char *format`
: a printf style string

`...`
: objects to print

Description

This function prints a user supplied message to the `tracefct` output stream, prefixed by the standard `tracefct` style prefix string. The message is passed in the same fashion as the arguments to `printf`, allowing formatted output. Unlike `tf_exit`, a newline character is *not* appended to the message.

3.8 tf_vmessage

Print a message to `stderr` using a `stdargs` argument list.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_vmessage(
  const char *format,
  va_list args
);
```

Parameters

`const char *format`
: a printf style string

`va_list args`
: objects to print

Description

This function prints a user supplied message to `stderr`, prefixed by the standard `tracefct` style prefix string. The error message is passed in the same fashion as the arguments to `vprintf`. Note that `vmessage` expects a *va_list* to be passed, rather than a variable argument list. Unlike `tf_exit`, a newline character is *not* appended to the message.

3.9 tf_dump_stack

Dump the called function stack

Synopsis

```
#include <tracefct/tracefct.h>

void tf_dump_stack(void);
```

Description

This function dumps the current function stack

3.10 tf_close

rest the `tracefct` output stream to `stderr`.

Synopsis

```
#include <tracefct/tracefct.h>

void tf_close(void);
```

Description

`tf_close` resets the output stream to `stderr`, closing the file previously opened by `tf_open`.

3.11 tf_open

redirect the `tracefct` output stream

Synopsis

```
#include <tracefct/tracefct.h>

int tf_open(const char *file);
```

Parameters

`const char *file`
: the file to which messages are to be written

Description

`tf_open` serves to change the `tracefct` output stream to the specified file. Normally `tracefct` writes to `stderr`. `tf_exit` *always* writes to `stderr` (see Section 3.6 [tf_exit], page 7). To reset the output stream, see Section 3.10 [tf_close], page 9. If the current output stream is not `stderr`, it is automatically closed.

If the passed filename is the string '`stderr`', `tf_open` will use `stderr`.

Returns

It returns zero upon success, non-zero upon failure.