NAME

TNF_PROBE, TNF_PROBE_0, TNF_PROBE_1, TNF_PROBE_2, TNF_PROBE_3, TNF_PROBE_4, TNF_PROBE_5 – probe insertion interface

SYNOPSIS

cc [*flag* ...] *file* ... [**–ltnfprobe**] [*library* ...]

#include <tnf/probe.h>

TNF_PROBE_0(*name***,** *keys***,** *detail***);**

TNF_PROBE_1(*name***,** *keys***,** *detail***,**
    *arg_type_1***,** *arg_name_1***,** *arg_value_1***);**

TNF_PROBE_2(*name***,** *keys***,** *detail***,**
    *arg_type_1***,** *arg_name_1***,** *arg_value_1***,**
    *arg_type_2***,** *arg_name_2***,** *arg_value_2***);**

TNF_PROBE_3(*name***,** *keys***,** *detail***,**
    *arg_type_1***,** *arg_name_1***,** *arg_value_1***,**
    *arg_type_2***,** *arg_name_2***,** *arg_value_2***,**
    *arg_type_3***,** *arg_name_3***,** *arg_value_3***);**

TNF_PROBE_4(*name***,** *keys***,** *detail***,**
    *arg_type_1***,** *arg_name_1***,** *arg_value_1***,**
    *arg_type_2***,** *arg_name_2***,** *arg_value_2***,**
    *arg_type_3***,** *arg_name_3***,** *arg_value_3***,**
    *arg_type_4***,** *arg_name_4***,** *arg_value_4***);**

TNF_PROBE_5(*name***,** *keys***,** *detail***,**
    *arg_type_1***,** *arg_name_1***,** *arg_value_1***,**
    *arg_type_2***,** *arg_name_2***,** *arg_value_2***,**
    *arg_type_3***,** *arg_name_3***,** *arg_value_3***,**
    *arg_type_4***,** *arg_name_4***,** *arg_value_4***,**
    *arg_type_5***,** *arg_name_5***,** *arg_value_5***);**

AVAILABILITY

SUNWtnfd

MT-LEVEL

MT-Safe.

DESCRIPTION

This macro interface is used to insert probes into C or C++ code. Probes can be placed anywhere in these programs including .init sections, .fini sections, multi-threaded code, shared objects, and shared objects opened by **dlopen**(3X). Probes can be used to generate trace data for performance analysis or to write debugging output to stderr. Probes are controlled by **prex**(1).

The trace data is logged to a trace file in Trace Normal Form (TNF) — the interface for the user to specify the name and size of the trace file is describe in **prex**(1). The trace file can be thought of as a least recently used circular buffer. Once the file has been filled, newer events will overwrite the older ones.

Compiling with the preprocessor option –**DNPROBE** (see **cc**(1B)), or with the preprocessor control statement **#define NPROBE** ahead of the **#include <tnf/probe.h>** statement, will stop probes from being compiled into the program.

name *name* of the probe and should follow the syntax guidelines for identifiers in ANSI C. The use of *name* declares it—hence no separate declaration is necessary. This is a block scope declaration, so it does not affect the name space of the program.

keys *keys* is a string of space separated keywords that specify the groups that the probe belongs to. A semicolon or single quotation mark is not allowed in this string. If any of the groups are enabled, the probe is enabled. *keys* cannot be a variable—it has to be an inlined string.

detail *detail* is a string that consists of <attribute> <value> pairs that are each separated by a semicolon. The first word (up to a space) is considered to be the attribute and the rest of the string (up to the semicolon) is considered the value. The value is optional. Semicolons or single quotation marks are not allowed in either the attribute or the value. *detail* is used for two reasons—first, it can be used to supply an attribute that a user can type into **prex**(1) to select probes. For example, if a user defines an attribute called color, then **prex**(1) can select probes based on the value of color. Secondly, *detail* can be used to annotate a probe with a string that is written out to a trace file only once. **prex**(1) uses spaces to tokenize the value when searching for a match. Spaces around the semicolon delimiter are allowed. *detail* cannot be a variable—it has to be an inlined string. For example, the *detail* string:

"sunw%debug entering function A; comX%exception no file; comY%func_entry; comY%color red blue"

consists of 4 units:

attribute	value	values that prex matches on
sunw%debug	entering function A	entering <or> function <or> A
comX%exception	no file	no <or> file
comY%func_entry		/.*/ (regular expression)
comY%color	red blue	red <or> blue

Attribute names have to be prefixed by the vendor stock symbol followed by the '%' character. This is to avoid collisions in the attribute name space. All attributes that do not have a '%' character are reserved. These are the predefined attributes:

attribute	semantics
name	name of probe
keys	keys of the probe (value is space separated tokens)
file	file name of the probe
line	line number of the probe
slots	slot names of the probe event (i.e., *arg_name_n*)
enable	off => probe is disabled. on => probe is enabled.
trace	off => tracing is off. on => tracing is on.
object	the executable or shared object that this probe is in.

funcs	list of probe functions connected to this probe.

arg_type_n

This is the type of the *n*th argument. These are the predefined TNF types:

tnf type	**associated C type (and semantics)**
tnf_long	int, long
tnf_ulong	unsigned int, unsigned long
tnf_longlong	long long (if implemented in compilation system)
tnf_ulonglong	unsigned long long (if implemented in compilation system)
tnf_float	float
tnf_double	double
tnf_string	char *
tnf_opaque	void *

To define new TNF types that are records consisting of the predefined TNF types or references to other user defined types, use the interface specified in **TNF_DECLARE_RECORD**(3X).

arg_name_n

arg_name_n is the name that the user wants associated with the *n*th argument. It should not have any quotes around it and should follow the syntax guidelines for identifiers in ANSI C. The string version of *arg_name_n* is stored for every probe and can be accessed as the attribute 'slots' as mentioned above.

arg_value_n

arg_value_n is evaluated to yield a value to be included in the trace file. A read access is done on any variables that are in mentioned in *arg_value_n*. In a multi-threaded program, it is the user's responsibility to place locks around the **TNF_PROBE** macro if *arg_value_n* contains a variable that needs to be read protected.

EXAMPLES

In this example, probes are placed at the entry and exit of a function to determine how much time is spent in the function. The function entry probe also logs the arguments to the function. When a probe is encountered at run time and if it is enabled for tracing, a record is generated to a trace file. All probes log the time when it was encountered and also have a reference to a tag record which has information like the file name, line number, *name*, *keys*, and *detail* of the probe. The first probe *work_args* will also log the value of the two arguments of the probe (*state* and *message*).

```
#include <tnf/probe.h>
int
work(int state, char *message)
{
      TNF_PROBE_2(work_start, "work_module work",
            "sunw%debug in function work",
            tnf_long,   int_input,    state,
            tnf_string, string_input, message);
         . . .
      TNF_PROBE_0(work_end, "work_module work", "");
}
```

SEE ALSO

cc(1B), **ld**(1), **prex**(1), **tnfdump**(1), **dlopen**(3X), **TNF_DECLARE_RECORD**(3X), **tnf_process_disable**(3X)

NOTES

If attaching to a running program with **prex**(1) to control the probes, compile the program with –**ltnfprobe** or start the program with the environment variable **LD_PRELOAD** set to libtnfprobe.so.1 (see **ld**(1)). If **libtnfprobe** is explicitly linked in to the program, it has to be before **libthread** on the link line.