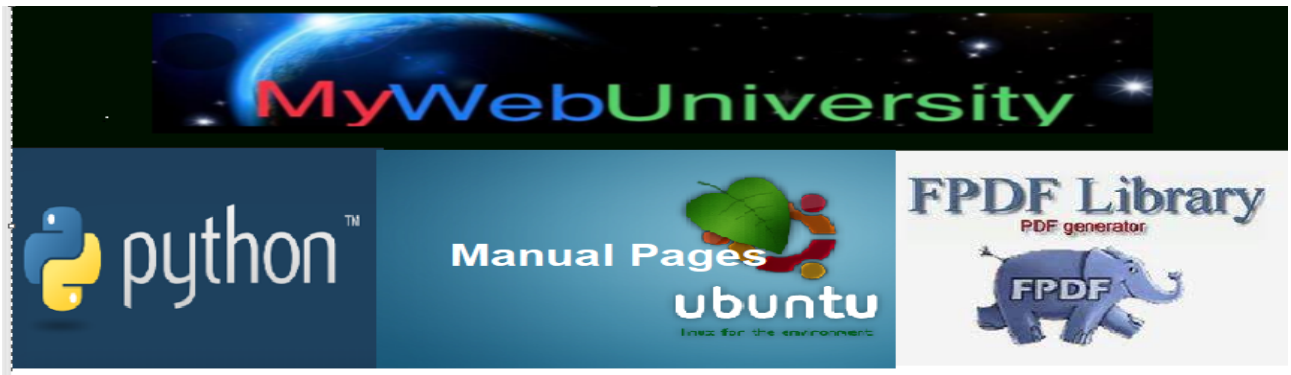

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'tracecmd_open.3'

$ man tracecmd_open.3

LIBTRACECMD(3) libtracefs Manual LIBTRACECMD(3)

NAME

tracecmd_open, tracecmd_open_fd, tracecmd_open_head, tracecmd_init_data, tracecmd_close, tracecmd_set_private, tracecmd_get_private - Open and close a trace file.

SYNOPSIS

```
#include <trace-cmd.h>

struct tracecmd_input *tracecmd_open(const char *file, int flags);
struct tracecmd_input *tracecmd_open_fd(int fd, int flags);
struct tracecmd_input *tracecmd_open_head(const char *file, int flags);
int tracecmd_init_data(struct tracecmd_input *handle);
void tracecmd_close(struct tracecmd_input *handle);
void tracecmd_set_private(struct tracecmd_input *handle, void *data);
void *tracecmd_get_private(struct tracecmd_input *handle);
```

DESCRIPTION

This set of APIs can be used to open and close a trace file recorded by trace-cmd(1) and containing tracing information from ftrace, the official Linux kernel tracer. The opened file is represented by a tracecmd_input structure, all other library APIs that work with the file require a pointer to the structure. The APIs for opening a trace file have a flag input parameter, which controls how the file will be opened and parsed. The flag is a combination

of these options:

TRACECMD_FL_LOAD_NO_PLUGINS - Do not load any plugins

TRACECMD_FL_LOAD_NO_SYSTEM_PLUGINS - Do not load system wide plugins, load only "local only" plugins from user's home directory.

The tracecmd_open() function opens a given trace file, parses the metadata headers from the file, allocates and initializes ? tracecmd_input handler structure representing the file. It also initializes the handler for reading trace data from the file. The returned handler is ready to be used with tracecmd_read_ APIs.

The tracecmd_open_fd() function does the same as tracecmd_open(), but works with a file descriptor to a trace file, opened for reading.

The tracecmd_open_head() function is the same as tracecmd_open(), but does not initialize the handler for reading trace data. It reads and parses the metadata headers only. The tracecmd_init_data() should be used before using the tracecmd_read_ APIs.

The tracecmd_init_data() function initializes a handle, allocated with tracecmd_open_head(), for reading trace data from the file associated with it. This API must be called before any of the tracecmd_read_ APIs.

The tracecmd_close() function frees a handle, pointer to tracecmd_input structure, previously allocated with tracecmd_open(), tracecmd_open_fd() or tracecmd_open_head() APIs.

The tracecmd_set_private() function allows to add specific data to the handle that can be retrieved later.

The tracecmd_get_private() function will retrieve the data set by tracecmd_set_private() for the given handle.

RETURN VALUE

The tracecmd_open(), tracecmd_open_fd() and tracecmd_open_head() functions return a pointer

to tracecmd_input structure or NULL in case of an error. The returned structure must be free with tracecmd_close(). Note that if tracecmd_open_fd() is used to allocate a tracecmd_input handler, when tracecmd_close() is called to close it, that fd will be closed also.

The tracecmd_init_data() function returns -1 in case of an error or 0 otherwise.

The tracecmd_get_private() returns the data set by tracecmd_set_private().

EXAMPLE

The are two different use patterns for opening and reading trace data from a trace file, which can be used depending on the use case.

1. Open and initialise the trace file in ? single step:

```
#include <stdlib.h>
#include <trace-cmd.h>

static int print_events(struct tracecmd_input *handle, struct tep_record *record, int cpu, void *data)
{
        static struct trace_seq seq;
        struct tep_handle *tep = tracecmd_get_tep(handle);
        const char *file = tracecmd_get_private(handle);

        if (!seq.buffer)
                trace_seq_init(&seq);

        trace_seq_reset(&seq);
        trace_seq_printf(&seq, "%s: ", file);
        tep_print_event(tep, &seq, record, "%6.1000d [%03d] %s-%d %s: %s\n",
                        TEP_PRINT_TIME, TEP_PRINT_CPU, TEP_PRINT_COMM, TEP_PRINT_PID,
                        TEP_PRINT_NAME, TEP_PRINT_INFO);
        trace_seq_terminate(&seq);
        trace_seq_do_printf(&seq);
```

```
	return 0;
}

int main(int argc, char **argv)
{
	struct tracecmd_input *handle;

	if (argc < 2) {
		printf("usage: %s trace.dat\n", argv[0]);
		exit(-1);
	}

	handle = tracecmd_open(argv[i], 0);
	if (!handle)
		exit(-1);

	tracecmd_set_private(handles[nr_handles], argv[i]);
	tracecmd_iterate_events(handles, NULL, 0, print_events, NULL);

	tracecmd_close(handle);
}
```

2. Open and initialise the trace file in two steps. This allows to perform
some processing based on metadata, read from the file, before initialising
the trace data for reading. Example for such use case is when opening multiple
trace files recorded in a same trace session. In that case timestamps of all
trace events must be adjusted based on the information from the file's metadata
and before reading the trace data.

```
#include <trace-cmd.h>
...
struct tracecmd_input *handle = tracecmd_open_head("trace.dat");
	if (!handle) {
```

```
        /* Failed to open trace.dat file */
    }
...
    /* do some processing, before initialising the trace data for reading */
...
    if (tracecmd_init_data(handle) < 0) {
        /* Failed to initialize hadle for reading the trace data */
    }
...
    /* Read tracing data from the file, using the handle */
...
    tracecmd_close(handle);
...
```

FILES

trace-cmd.h

Header file to include in order to have access to the library APIs.

-ltracecmd

Linker switch to add when building a program that uses the library.

SEE ALSO

libtracefs(3), libtraceevent(3), trace-cmd(1) trace-cmd.dat(5)

AUTHOR

Steven Rostedt <rostedt@goodmis.org[1]>

Tzvetomir Stoyanov <tz.stoyanov@gmail.com[2]>

REPORTING BUGS

Report bugs to <linux-trace-devel@vger.kernel.org[3]>

LICENSE

RESOURCES

https://git.kernel.org/pub/scm/utils/trace-cmd/trace-cmd.git/

COPYING

NOTES

1. rostedt@goodmis.org

   mailto:rostedt@goodmis.org

2. tz.stoyanov@gmail.com

   mailto:tz.stoyanov@gmail.com

3. linux-trace-devel@vger.kernel.org

   mailto:linux-trace-devel@vger.kernel.org