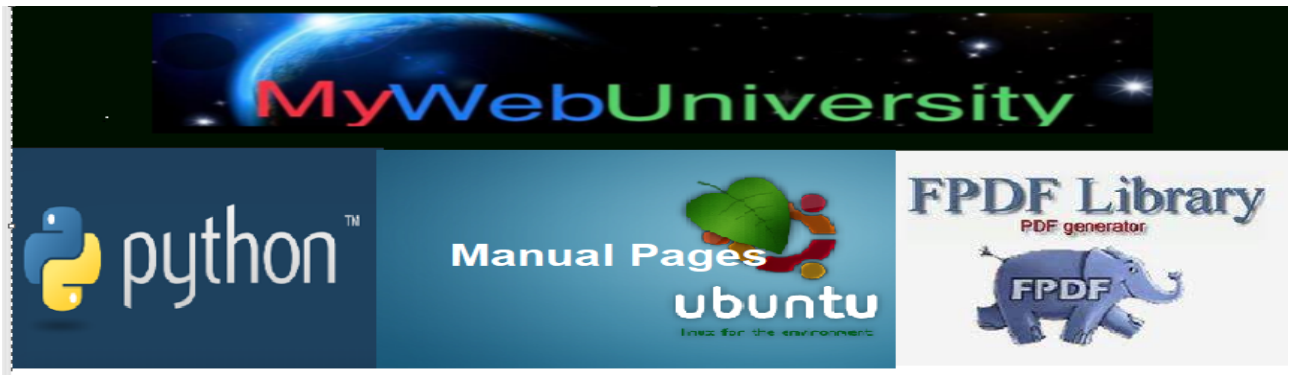

Full credit is given to the above companies including the OS that this PDF file was generated!

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

$ man tracecmd_map_get_guest.3

LIBTRACECMD(3) libtracefs Manual LIBTRACECMD(3)

NAME

tracecmd_map_vcpus, tracecmd_get_cpu_map, tracecmd_map_find_by_host_pid, tracecmd_map_get_host_pid, tracecmd_map_get_guest, tracecmd_map_set_private, tracecmd_map_get_private - Mapping host and guest data

SYNOPSIS

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

int tracecmd_map_vcpus(struct tracecmd_input **handles, int nr_handles);
struct tracecmd_cpu_map *tracecmd_get_cpu_map(struct tracecmd_input *handle, int cpu);
struct tracecmd_cpu_map *tracecmd_map_find_by_host_pid(struct tracecmd_input *handle,
                                                       int host_pid);
int tracecmd_map_get_host_pid(struct tracecmd_cpu_map *map);
struct tracecmd_input *tracecmd_map_get_guest(struct tracecmd_cpu_map *map);
void tracecmd_map_set_private(struct tracecmd_cpu_map *map, void *priv);
void *tracecmd_map_get_private(struct tracecmd_cpu_map *map);
```

DESCRIPTION

This set of APIs is used to map host and guest trace files for to facilitate further tracing analysis.

The tracecmd_map_vcpus() takes an array of handles where each item in that array was created by one of the tracecmd_open(3) functions, and the number of handles as nr_handles. The first handle in the array of handles is expected to be the descriptor for the host tracing file, and the rest are guest trace files that run on the host, and were created by the trace-cmd record(1) and trace-cmd agent(1) interactions. It returns the number of guests found in handles that were associated with the host, or negative on error.

The tracecmd_get_cpu_map() returns a descriptor for a given CPU for a handle. If the handle was a guest defined from tracecmd_map_vcpus() then the mapping created from that function that is associated to this particular vCPU (denoted by cpu) from handle. This destriptor can be used by tarcecmd_map_get_guest(), tracecmd_map_set_private() and tracecmd_map_get_private() functions.

The tracecmd_map_find_by_host_pid() will return a mapping for a guest virtual CPU that is handled by the given host_pid. Note, the handle passed in can be either the host handle or one of the guest?s handles for that host that was mapped by tracecmd_map_vcpus(), even if the guest handle does not have the vCPU that the host_pid represents.

The tracecmd_map_get_host_pid() will recturn the host_pid for a given map that was retrieved by one of the above functions.

The tracecmd_map_get_guest() will recturn the guest_handle for a given map that was retrieved by one of the above functions.

The tracecmd_map_set_private() allows the application to assign private data for a given guest vCPU to host thread mapping defined by map.

The tracecmd_map_get_private() retrieves the priv data from map that was set by tracecmd_map_set_private().

RETURN VALUE

tracecmd_map_vcpus() returns the number of guests in the handles array that were mapped to the host handle that is the first entry in handles. It returns -1 on error.

tracecmd_get_cpu_map() returns a map created by tracecmd_map_vcpus() for a given cpu for a given handle, or NULL if it is not found.

tracecmd_map_find_by_host_pid() returns a map that is associated by the host task with host_pid as its process ID. handle can be either a the host handle, or one of the guest handles that were mapped to the host via tracecmd_map_vcpus(), even if the guest handle is another guest than the one that the mapping is for. It returns NULL if not found.

tracecmd_map_get_host_pid() returns the host process ID for an associated mapping defined by map.

tracecmd_map_get_guest() returns the guest handle for an associated mapping defined by map.

tracecmd_map_get_private() returns the private data of a mapping defined by map that was set by tracecmd_map_set_private().

EXAMPLE

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

int main(int argc, char **argv)
{
        struct tracecmd_input **handles = NULL;
        int nr_handles;
        int i;

        if (argc < 2) {
                printf("usage: host_trace.dat guest1_trace.dat [guest2_trace.dat ...]\n");
                exit(-1);
        }
```

```
for (i = 1; i < argc; i++) {
        handles = realloc(handles, sizeof(*handles) * (nr_handles + 1));
        if (!handles)
                exit(-1);
        handles[nr_handles] = tracecmd_open(argv[i], 0);
        if (!handles[nr_handles]) {
                perror(argv[1]);
                exit(-1);
        }
        tracecmd_set_private(handles[nr_handles], argv[i]);
        nr_handles++;
}

tracecmd_map_vcpus(handles, nr_handles);

for (i = 1; i < nr_handles; i++) {
        struct tracecmd_cpu_map *map;
        struct tep_handle *tep;
        const char *file = tracecmd_get_private(handles[i]);
        int cpus, cpu;

        printf("Mappings for guest %s:\n", file);
        tep = tracecmd_get_tep(handles[i]);
        cpus = tep_get_cpus(tep);
        for (cpu = 0; cpu < cpus; cpu++) {
                printf("  [%03d] ", cpu);
                map = tracecmd_get_cpu_map(handles[i], cpu);
                if (!map) {
                        printf("Has no mapping!\n");
                        continue;
                }
                printf("host_pid: %d\n", tracecmd_map_get_host_pid(map));
        }
```

```
        }
        for (i = 0; i < nr_handles; i++)
                tracecmd_close(handles[i]);
        free(handles);
        exit(0);
}
```

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)

REPORTING BUGS

Report bugs to <linux-trace-devel@vger.kernel.org[1]>

LICENSE

libtracecmd is Free Software licensed under the GNU LGPL 2.1

RESOURCES

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

COPYING

NOTES

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

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