**NAME**

tnfdump – converts binary TNF file to ASCII

**SYNOPSIS**

**tnfdump** [ **–r** ] *tnf_file* . . .

**AVAILABILITY**

SUNWtnfd

**DESCRIPTION**

**tnfdump** converts the specified binary TNF trace files to ASCII. The ASCII output can be used to do performance analysis. The default mode (without the -r option) prints all the event records (that were generated by **TNF_PROBE**(3X)) and the event descriptor records only. It also orders the events by time.

**OPTIONS**

**–r** Does a raw conversion of TNF to ASCII. The output is a literal transalation of the binary TNF file and includes all the records in the file. This output is useful only if you have a good understanding of TNF. A sample output is listed in **EXAMPLES** below.

**RETURN VALUES**

**tnfdump** returns **0** on succcessful exit.

**EXAMPLES**

To convert the file **/tmp/trace-2130** into ASCII use:

**example% tnfdump /tmp/trace-2130**

```
probe     tnf_name: "inloop" tnf_string: "keys cookie main loop;file cookie2.c;line 50;sunw%debug in the loop"
probe     tnf_name: "end" tnf_string: "keys cookie main end;file cookie2.c;line 41;sunw%debug exiting program"
---------------- ---------------- ----- ----- ---------- --- ------------------------- ------------------------
  Elapsed (ms)     Delta (ms)   PID LWPID   TID   CPU Probe Name        Data / Description . . .
---------------- ---------------- ----- ----- ---------- --- ------------------------- ------------------------
     0.000000       0.000000   8792   1       0    - inloop            loop_count: 0 total_iterations: 0
     0.339000       0.339000   8792   1       0    - inloop            loop_count: 1 total_iterations: 1
     0.350500       0.011500   8792   1       0    - inloop            loop_count: 2 total_iterations: 2
     0.359500       0.009000   8792   1       0    - inloop            loop_count: 3 total_iterations: 3
     0.369500       0.010000   8792   1       0    - inloop            loop_count: 4 total_iterations: 4
  7775.969500    7775.600000   8792   1       0    - inloop            loop_count: 0 total_iterations: 5
  7776.016000       0.046500   8792   1       0    - inloop            loop_count: 1 total_iterations: 6
  7776.025000       0.009000   8792   1       0    - inloop            loop_count: 2 total_iterations: 7
  7776.034000       0.009000   8792   1       0    - inloop            loop_count: 3 total_iterations: 8
  7776.043000       0.009000   8792   1       0    - inloop            loop_count: 4 total_iterations: 9
  7776.052000       0.009000   8792   1       0    - inloop            loop_count: 5 total_iterations: 10
  7776.061000       0.009000   8792   1       0    - inloop            loop_count: 6 total_iterations: 11
  9475.979500    1699.918500   8792   1       0    - end               node_struct: { type: node_tnf
                                                                   cur_sum: 9 max_cnt: 12 }
```

All probes that are encountered during execution have a description of it printed out. The description is one per line prefixed by the keyword **'probe'.** The name of the probe is in double quotes after the keyword **'tnf_name'.** The description of this probe is in double quotes after the keyword **'tnf_string'.**

A heading is printed after all the description of the probes are printed. The first column gives the elapsed time in milli-seconds since the first event. The second column gives the elapsed time in milli-seconds since the previous event. The next four columns are the process id, lwp id, thread id, and cpu number. The next column is the name of the probe that generated this event. This can be matched to the probe description explained above. The last column is the data that the event contains formatted as **arg_name_n (see TNF_PROBE**(3X)) followed by a colon and the value of that argument. The format of the value depends on its type — **tnf_opaque** arguments are printed in hex, all other integers are printed in decimal, strings are printed in double quotes, and user defined records are enclosed in braces '{ }'. The first field of a user defined record indicates its TNF type (see **TNF_DECLARE_RECORD**(3X)) and the rest of the fields are the members of the record.

A '-' in any column indicates that there is no data for that particular column.

To do a raw conversion of the file **/tmp/trace-4000** into ASCII use:

```
example% tnfdump -r /tmp/trace-4000
```

The output will look like the following:

```
0x10e00   : {
              tnf_tag 0x109c0    tnf_block_header
           generation 1
          bytes_valid 320
               A_lock 0
               B_lock 0
           next_block 0x0
        }
0x10e10   : {
              tnf_tag 0x10010    probe1
          tnf_tag_arg 0x10e24    <tnf_sched_rec>
           time_delta 128
           test_ulong 4294967295
            test_long -1
        }
0x10e24   : {
              tnf_tag 0x10cf4    tnf_sched_rec
                  tid 0
                lwpid 1
                  pid 13568
            time_base 277077875828500
        }
0x10e3c   : {
              tnf_tag 0x11010    probe2
          tnf_tag_arg 0x10e24    <tnf_sched_rec>
           time_delta 735500
             test_str 0x10e48    "string1"
        }
0x10e48   : {
```

```
                tnf_tag 0x1072c    tnf_string
          tnf_self_size 16
                  chars "string1"
        }
0x10e58   : {
                tnf_tag 0x110ec    probe3
            tnf_tag_arg 0x10e24    <tnf_sched_rec>
             time_delta 868000
         test_ulonglong 18446744073709551615
          test_longlong -1
             test_float 3.142857
        }
0x110ec   : {
                tnf_tag 0x10030    tnf_probe_type
           tnf_tag_code 42
               tnf_name 0x1110c    "probe3"
         tnf_properties 0x1111c    <tnf_properties>
         tnf_slot_types 0x11130    <tnf_slot_types>
          tnf_type_size 32
         tnf_slot_names 0x111c4    <tnf_slot_names>
             tnf_string 0x11268    "keys targdebug main;file targdebug.c;line 6
1;"
        }
0x1110c   : {
                tnf_tag 0x10068    tnf_name
          tnf_self_size 16
                  chars "probe3"
        }
0x1111c   : {
                tnf_tag 0x100b4    tnf_properties
          tnf_self_size 20
                      0 0x101a0    tnf_tagged
                      1 0x101c4    tnf_struct
                      2 0x10b84    tnf_tag_arg
        }
0x11130   : {
                tnf_tag 0x10210    tnf_slot_types
          tnf_self_size 28
                      0 0x10bd0    tnf_probe_event
                      1 0x10c20    tnf_time_delta
                      2 0x1114c    tnf_uint64
                      3 0x10d54    tnf_int64
                      4 0x11188    tnf_float32
        }
```

The first number is the file offset of the record. The record is enclosed in braces '{ }'. The first column in a record is the slot name (for records whose fields do not have names, it is the type name). The second column in the record is the value of that slot if it is a scalar (only scalars that are of type **tnf_opaque** are printed in hex), or the offset of the record if it is a reference to another record.

The third column in a record is optional. It does not exist for scalar slots of records. If it exists, the third column is a type name with or without angle brackets, or a string in double quotes. Unadorned names indicate a reference to the named metatag record (i.e. a reference to a record with that name in the **tnf_name** field). Type names in angled brackets indicate a reference to a record that is an instance of that type (i.e., a reference to a record with that name in the **tnf_tag** field). The content of strings are printed out in double quotes at the reference site.

Records that are arrays have their array elements follow the header slots, and are numbered 0, 1, 2, etc., except strings where the string is written as the 'chars' (pseudo-name) slot.

Records that are events (generated by **TNF_PROBE**(3X)) will have a slot name of **tnf_tag_arg** as their second field which is a reference to the schedule record. Schedule records describe more information about the event like the thread-id, process-id, and the **time_base.** The **time_delta** of an event can be added to the **time_base** of the schedule record that the event references, to give an absolute time. This time is expressed as nanoseconds since some arbitrary time in the past (see **gethrtime**(3C)).

## SEE ALSO

**prex**(1), **gethrtime**(3C), **TNF_DECLARE_RECORD**(3X), **TNF_PROBE**(3X), **tnf_process_disable**(3X)