NAME

ktrace - enable kernel process tracing

SYNOPSIS

ktrace [**-aCcdi**] [**-f** *trfile*] [**-g** *pgrp* | **-p** *pid*] [**-t** *trstr*]
ktrace [**-adi**] [**-f** *trfile*] [**-t** *trstr*] *command*

DESCRIPTION

The **ktrace** utility enables kernel trace logging for the specified processes. Kernel trace data is logged to the file *ktrace.out*. The kernel operations that are traced include system calls (see intro(2)), file system path lookups (namei(9)), signal processing (sigaction(2)), and I/O.

Once tracing is enabled on a process, trace data will be logged until either the process exits or the trace point is cleared. A traced process can generate enormous amounts of log data quickly; It is strongly suggested that users memorize how to disable tracing before attempting to trace a process. The following command is sufficient to disable tracing on all user-owned processes, and, if executed by root, all processes:

```
$ ktrace -C
```

The trace file is not human readable; use kdump(1) to decode it.

The utility may be used only with a kernel that has been built with the "KTRACE" option in the kernel configuration file.

The options are:

-a Append to the trace file instead of recreating it.

-C Disable tracing on all user-owned processes, and, if executed by root, all processes in the system.

-c Clear the specified trace points associated with the given file or processes.

-d Descendants; perform the operation for all current children of the designated processes. See also the **-i** option.

-f *trfile*
Log trace records to *trfile* instead of *ktrace.out*.

-g *pgid*

Enable (disable) tracing on all processes in the process group (only one **-g** flag is permitted).

-i Inherit; pass the trace flags to all future children of the designated processes. See also the **-d** option.

-p *pid* Enable (disable) tracing on the indicated process id (only one **-p** flag is permitted).

-t *trstr* Specify the list of trace points to enable or disable, one per letter. If an explicit list is not specified, the default set of trace points is used.

The following trace points are supported:

- **c** trace system calls
- **f** trace page faults
- **i** trace I/O
- **n** trace namei(9) translations
- **p** trace capability check failures
- **s** trace signal processing
- **t** trace various structures
- **u** userland traces generated by utrace(2)
- **w** context switches
- **y** trace sysctl(3) requests
- **+** trace the default set of trace points - **c**, **i**, **n**, **s**, **t**, **u**, **y**

command

Execute *command* with the specified trace flags.

The **-p**, **-g**, and *command* options are mutually exclusive.

CAPABILITY VIOLATION TRACING

When the **p** trace point is specified, **ktrace** will record capsicum(4) capability mode violations made by the traced process. Violations will be logged regardless of whether the process has actually entered capability mode.

For developers that are interested in Capsicumizing their programs, the **c**, **n**, **p** trace points can help quickly identify any system calls and path lookups that are triggering violations.

EXAMPLES

Run "make", then trace it and any child processes:

```
$ ktrace -i make
```

Trace all kernel operations of process id 34:

```
$ ktrace -p 34
```

Trace all kernel operations of processes in process group 15 and pass the trace flags to all current and future children:

```
$ ktrace -idg 15
```

Disable all tracing of process 65:

```
$ ktrace -cp 65
```

Disable tracing signals on process 70 and all current children:

```
$ ktrace -t s -cdp 70
```

Enable tracing of I/O on process 67:

```
$ ktrace -ti -p 67
```

Disable all tracing to the file "tracedata":

```
$ ktrace -c -f tracedata
```

Disable tracing of all user-owned processes:

```
$ ktrace -C
```

SEE ALSO

dtrace(1), kdump(1), truss(1), intro(2), ktrace(2), sigaction(2), utrace(2), capsicum(4), namei(9)

HISTORY

The **ktrace** command appeared in 4.4BSD.

BUGS

Only works if *trfile* is a regular file.