NAME

ktrace - process tracing

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/param.h>
#include <sys/time.h>
#include <sys/uio.h>
#include <sys/ktrace.h>

int
ktrace(*const char *tracefile, int ops, int trpoints, int pid*);

DESCRIPTION

The **ktrace**() system call enables or disables tracing of one or more processes. Users may only trace their own processes. Only the super-user can trace setuid or setgid programs.

The *tracefile* argument gives the pathname of the file to be used for tracing. The file must exist and be a regular file writable by the calling process. All trace records are always appended to the file, so the file must be truncated to zero length to discard previous trace data. If tracing points are being disabled (see KTROP_CLEAR below), *tracefile* may be NULL.

The *ops* argument specifies the requested ktrace operation. The defined operations are:

KTROP_SET	Enable trace points specified in *trpoints*.
KTROP_CLEAR	Disable trace points specified in *trpoints*.
KTROP_CLEARFILE	Stop all tracing.
KTRFLAG_DESCEND	The tracing change should apply to the specified process and all its current children.

The *trpoints* argument specifies the trace points of interest. The defined trace points are:

KTRFAC_SYSCALL	Trace system calls.
KTRFAC_SYSRET	Trace return values from system calls.
KTRFAC_NAMEI	Trace name lookup operations.
KTRFAC_GENIO	Trace all I/O (note that this option can generate much output).
KTRFAC_PSIG	Trace posted signals.
KTRFAC_CSW	Trace context switch points.

KTRFAC_USER	Trace application-specific events.
KTRFAC_STRUCT	Trace certain data structures.
KTRFAC_SYSCTL	Trace sysctls.
KTRFAC_PROCCTOR	Trace process construction.
KTRFAC_PROCDTOR	Trace process destruction.
KTRFAC_CAPFAIL	Trace capability failures.
KTRFAC_FAULT	Trace page faults.
KTRFAC_FAULTEND	Trace the end of page faults.
KTRFAC_STRUCT_ARRAY	Trace arrays of certain data structures.
KTRFAC_INHERIT	Inherit tracing to future children.

Each tracing event outputs a record composed of a generic header followed by a trace point specific structure. The generic header is:

```
struct ktr_header {
        int             ktr_len;        /* length of buf */
        short           ktr_type;       /* trace record type */
        pid_t           ktr_pid;        /* process id */
        char            ktr_comm[MAXCOMLEN+1];      /* command name */
        struct timeval  ktr_time;       /* timestamp */
        long            ktr_tid;        /* thread id */
};
```

The *ktr_len* field specifies the length of the *ktr_type* data that follows this header. The *ktr_pid* and *ktr_comm* fields specify the process and command generating the record. The *ktr_time* field gives the time (with microsecond resolution) that the record was generated. The *ktr_tid* field holds a thread id.

The generic header is followed by *ktr_len* bytes of a *ktr_type* record. The type specific records are defined in the <*sys/ktrace.h*> include file.

SYSCTL TUNABLES

The following sysctl(8) tunables influence the behaviour of **ktrace**():

kern.ktrace.genio_size
: bounds the amount of data a traced I/O request will log to the trace file.

kern.ktrace.request_pool
: bounds the number of trace events being logged at a time.

Sysctl tunables that control process debuggability (as determined by p_candebug(9)) also affect the

operation of **ktrace**().

RETURN VALUES

The **ktrace**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **ktrace**() system call will fail if:

[ENOTDIR] A component of the path prefix is not a directory.

[ENAMETOOLONG] A component of a pathname exceeded 255 characters, or an entire path name exceeded 1023 characters.

[ENOENT] The named tracefile does not exist.

[EACCES] Search permission is denied for a component of the path prefix.

[ELOOP] Too many symbolic links were encountered in translating the pathname.

[EIO] An I/O error occurred while reading from or writing to the file system.

[EINTEGRITY] Corrupted data was detected while reading from the file system.

[ENOSYS] The kernel was not compiled with **ktrace** support.

A thread may be unable to log one or more tracing events due to a temporary shortage of resources. This condition is remembered by the kernel, and the next tracing request that succeeds will have the flag KTR_DROP set in its *ktr_type* field.

SEE ALSO

kdump(1), ktrace(1), utrace(2), sysctl(8), p_candebug(9)

HISTORY

The **ktrace**() system call first appeared in 4.4BSD.