NAME

CTR0, **CTR1**, **CTR2**, **CTR3**, **CTR4**, **CTR5** - kernel tracing facility

SYNOPSIS

#include <sys/param.h>
#include <sys/ktr.h>

extern int ktr_cpumask;
extern int ktr_entries;
extern int ktr_extend;
extern int ktr_mask;
extern int ktr_verbose;
extern struct ktr_entry ktr_buf[];

void
CTR(*u_int mask*, *char *format*, ...);

void
CTR0(*u_int mask*, *char *format*);

void
CTR1(*u_int mask*, *char *format*, *arg1*);

void
CTR2(*u_int mask*, *char *format*, *arg1*, *arg2*);

void
CTR3(*u_int mask*, *char *format*, *arg1*, *arg2*, *arg3*);

void
CTR4(*u_int mask*, *char *format*, *arg1*, *arg2*, *arg3*, *arg4*);

void
CTR5(*u_int mask*, *char *format*, *arg1*, *arg2*, *arg3*, *arg4*, *arg5*);

void
CTR6(*u_int mask*, *char *format*, *arg1*, *arg2*, *arg3*, *arg4*, *arg5*, *arg6*);

DESCRIPTION

KTR provides a circular buffer of events that can be logged in a printf(9) style fashion. These events

can then be dumped with ddb(4), gdb(1) (*ports/devel/gdb*) or ktrdump(8).

Events are created and logged in the kernel via the CTR and CTR*x* macros. The first parameter is a mask of event types (KTR_*) defined in <*sys/ktr_class.h*>. The event will be logged only if any of the event types specified in *mask* are enabled in the global event mask stored in *ktr_mask*. The *format* argument is a printf(9) style format string used to build the text of the event log message. Following the *format* string are zero to six arguments referenced by *format*. Each event is logged with a file name and source line number of the originating CTR call, and a timestamp in addition to the log message.

The event is stored in the circular buffer with supplied arguments as is, and formatting is done at the dump time. Do not use pointers to the objects with limited lifetime, for instance, strings, because the pointer may become invalid when buffer is printed.

The CTR*x* macros differ only in the number of arguments each one takes, as indicated by its name.

The *ktr_entries* variable contains the number of entries in the *ktr_buf* array. These variables are mostly useful for post-mortem crash dump tools to locate the base of the circular trace buffer and its length.

The *ktr_mask* variable contains the run time mask of events to log.

The CPU event mask is stored in the *ktr_cpumask* variable.

The *ktr_verbose* variable stores the verbose flag that controls whether events are logged to the console in addition to the event buffer.

EXAMPLES

This example demonstrates the use of tracepoints at the KTR_PROC logging level.

```
void
mi_switch()
{
        ...
        /*
         * Pick a new current process and record its start time.
         */
        ...
        CTR3(KTR_PROC, "mi_switch: old proc %p (pid %d)", p, p->p_pid);
        ...
        cpu_switch();
        ...
```

```
        CTR3(KTR_PROC, "mi_switch: new proc %p (pid %d)", p, p->p_pid);
        ...
}
```

SEE ALSO

ktr(4), ktrdump(8)

HISTORY

The KTR kernel tracing facility first appeared in BSD/OS 3.0 and was imported into FreeBSD 5.0.

The **CTR**() macro accepting a variable number of arguments first appeared in FreeBSD 14.0.

BUGS

Currently there is one global buffer shared among all CPUs. It might be profitable at some point in time to use per-CPU buffers instead so that if one CPU halts or starts spinning, then the log messages it emitted just prior to halting or spinning will not be drowned out by events from the other CPUs.

The arguments given in **CTRx**() macros are stored as *u_long*, so do not pass arguments larger than size of an *u_long* type. For example passing 64bit arguments on 32bit architectures will give incorrect results.