NAME

kcov - interface for collecting kernel code coverage information

SYNOPSIS

To compile KCOV into the kernel, place the following lines in your kernel configuration file:

```
options COVERAGE
options KCOV
```

The following header file defines the application interface provided by KCOV:

```
#include <sys/kcov.h>
```

DESCRIPTION

kcov is a module that enables collection of code coverage information from the kernel. It relies on code instrumentation enabled by the COVERAGE kernel option. When **kcov** is enabled by a user-mode thread, it collects coverage information only for that thread, excluding hard interrupt handlers. As a result, **kcov** is not suited to collect comprehensive coverage data for the entire kernel; its main purpose is to provide input for coverage-guided system call fuzzers.

In typical usage, a user-mode thread first allocates a chunk of memory to be shared with **kcov**. The thread then enables coverage tracing, with coverage data being written by the kernel to the shared memory region. When tracing is disabled, the kernel relinquishes its access to the shared memory region, and the written coverage data may be consumed.

The shared memory buffer can be treated as a 64-bit unsigned integer followed by an array of records. The integer records the number of valid records and is updated by the kernel as coverage information is recorded. The state of the tracing buffer can be reset by writing the value 0 to this field. The record layout depends on the tracing mode set using the KIOENABLE ioctl.

Two tracing modes are implemented, KCOV_MODE_TRACE_PC and KCOV_MODE_TRACE_CMP. PC-tracing records a program counter value for each basic block executed while tracing is enabled. In this mode, each record is a single 64-bit unsigned integer containing the program counter value. Comparison tracing provides information about data flow; information about dynamic variable comparisons is recorded. Such records provide information about the results of c(7) ‘if’ or ‘switch’ statements, for example. In this mode each record consists of four 64-bit unsigned integers. The first integer is a bitmask defining attributes of the variables involved in the comparison. KCOV_CMP_CONST is set if one of the variables has a constant value at compile-time, and KCOV_CMP_SIZE(n) specifies the width of the variables:

KCOV_CMP_SIZE(0): a comparison of 8-bit integers
KCOV_CMP_SIZE(1): a comparison of 16-bit integers
KCOV_CMP_SIZE(2): a comparison of 32-bit integers
KCOV_CMP_SIZE(3): a comparison of 64-bit integers

The second and third fields record the values of the two variables, and the fourth and final field stores the program counter value of the comparison.

IOCTL INTERFACE

Applications interact with **kcov** using the ioctl(2) system call. Each thread making use of **kcov** must use a separate file descriptor for */dev/kcov*. The following ioctls are defined:

KIOSETBUFSIZE *size_t entries*
: Set the size of the tracing buffer in units of KCOV_ENTRY_SIZE. The buffer may then be mapped into the calling thread's address space by calling mmap(2) on the **kcov** device file.

KIOENABLE *int mode*
: Enable coverage tracing for the calling thread. Valid modes are KCOV_MODE_TRACE_PC and KCOV_MODE_TRACE_CMP.

KIODISABLE
: Disable coverage tracing for the calling thread.

FILES

kcov creates the */dev/kcov* device file.

EXAMPLES

The following code sample collects information about basic block coverage for kernel code executed while printing 'Hello, world'.

```
size_t sz;
uint64_t *buf;
int fd;

fd = open("/dev/kcov", O_RDWR);
if (fd == -1)
        err(1, "open(/dev/kcov)");
sz = 1ul << 20; /* 1MB */
if (ioctl(fd, KIOSETBUFSIZE, sz / KCOV_ENTRY_SIZE) != 0)
        err(1, "ioctl(KIOSETBUFSIZE)");
```

```
buf = mmap(NULL, sz, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
if (buf == MAP_FAILED)
        err(1, "mmap");

/* Enable PC tracing. */
if (ioctl(fd, KIOENABLE, KCOV_MODE_TRACE_PC) != 0)
        err(1, "ioctl(KIOENABLE)");

/* Clear trace records from the preceding ioctl() call. */
buf[0] = 0;

printf("Hello, world!\n");

/* Disable PC tracing. */
if (ioctl(fd, KIODISABLE, 0) != 0)
        err(1, "ioctl(KIODISABLE)");

for (uint64_t i = 1; i < buf[0]; i++)
        printf("%#jx\n", (uintmax_t)buf[i]);
```

The output of this program can be approximately mapped to line numbers in kernel source code:

```
# ./kcov-test | sed 1d | addr2line -e /usr/lib/debug/boot/kernel/kernel.debug
```

SEE ALSO

ioctl(2), mmap(2)

HISTORY

kcov first appeared in FreeBSD 13.0.

BUGS

The FreeBSD implementation of **kcov** does not yet support remote tracing.