```
#include <linux/uaccess.h>
```

```
/*
 * probe_kernel_read(): safely attempt to read from a location
 * @dst: pointer to the buffer that shall take the data
 * @src: address to read from
 * @size: size of the data chunk
 *
 * Safely read from address @src to the buffer at @dst.  If a kernel fault
 * happens, handle that and return -EFAULT.
 */
extern long probe_kernel_read(void *dst, const void *src, size_t size);

/*
 * probe_kernel_write(): safely attempt to write to a location
 * @dst: address to write to
 * @src: pointer to the data that shall be written
 * @size: size of the data chunk
 *
 * Safely write to address @dst from the buffer at @src.  If a kernel fault
 * happens, handle that and return -EFAULT.
 */
extern long notrace probe_kernel_write(void *dst, const void *src, size_t si
ze);
```

如果 `src` 所指向的memory不可读/不可写，则返回-EFAULT(而不会crash)

kernel debugger就充分利用了这个feature

```
int __weak kgdb_arch_set_breakpoint(struct kgdb_bkpt *bpt)
{
    int err;

    err = probe_kernel_read(bpt->saved_instr, (char *)bpt->bpt_addr,
                BREAK_INSTR_SIZE);
    if (err)
        return err;
    err = probe_kernel_write((char *)bpt->bpt_addr,
                 arch_kgdb_ops.gdb_bpt_instr, BREAK_INSTR_SIZE);
    return err;
}

int __weak kgdb_arch_remove_breakpoint(struct kgdb_bkpt *bpt)
{
    return probe_kernel_write((char *)bpt->bpt_addr,
                  (char *)bpt->saved_instr, BREAK_INSTR_SIZE);
}
```