```
/*
 * Exercise /dev/mem mmap cases that have been troublesome in the past
 *
 * (c) Copyright 2007 Hewlett-Packard Development Company, L.P.
 *  Bjorn Helgaas <bjorn.helgaas@hp.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 */

#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>
#include <fcntl.h>
#include <fnmatch.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <unistd.h>
#include <linux/pci.h>

int sum;

static int map_mem(char *path, off_t offset, size_t length, int touch)
{
    int fd, rc;
    void *addr;
    int *c;

    fd = open(path, O_RDWR);
    if (fd == -1) {
        perror(path);
        return -1;
    }

    if (fnmatch("/proc/bus/pci/*", path, 0) == 0) {
        rc = ioctl(fd, PCIIOC_MMAP_IS_MEM);
        if (rc == -1)
            perror("PCIIOC_MMAP_IS_MEM ioctl");
    }

    addr = mmap(NULL, length, PROT_READ|PROT_WRITE, MAP_SHARED, fd, offset);
    if (addr == MAP_FAILED)
        return 1;

    if (touch) {
        c = (int *) addr;
        while (c < (int *) (addr + length))
            sum += *c++;
    }

    rc = munmap(addr, length);
    if (rc == -1) {
        perror("munmap");
        return -1;
    }

    close(fd);
    return 0;
}

static int scan_tree(char *path, char *file, off_t offset, size_t length, int touch)
{
```

```
    struct dirent **namelist;
    char *name, *path2;
    int i, n, r, rc = 0, result = 0;
    struct stat buf;

    n = scandir(path, &namelist, 0, alphasort);
    if (n < 0) {
        perror("scandir");
        return -1;
    }

    for (i = 0; i < n; i++) {
        name = namelist[i]->d_name;

        if (fnmatch(".", name, 0) == 0)
            goto skip;
        if (fnmatch("..", name, 0) == 0)
            goto skip;

        path2 = malloc(strlen(path) + strlen(name) + 3);
        strcpy(path2, path);
        strcat(path2, "/");
        strcat(path2, name);

        if (fnmatch(file, name, 0) == 0) {
            rc = map_mem(path2, offset, length, touch);
            if (rc == 0)
                fprintf(stderr, "PASS: %s 0x%lx-0x%lx is %s\n", path2, offset, offset + length, touch
? "readable" : "mappable");
            else if (rc > 0)
                fprintf(stderr, "PASS: %s 0x%lx-0x%lx not mappable\n", path2, offset, offset +
length);
            else {
                fprintf(stderr, "FAIL: %s 0x%lx-0x%lx not accessible\n", path2, offset, offset +
length);
                return rc;
            }
        } else {
            r = lstat(path2, &buf);
            if (r == 0 && S_ISDIR(buf.st_mode)) {
                rc = scan_tree(path2, file, offset, length, touch);
                if (rc < 0)
                    return rc;
            }
        }

        result |= rc;
        free(path2);

skip:
        free(namelist[i]);
    }
    free(namelist);
    return result;
}

char buf[1024];

static int read_rom(char *path)
{
    int fd, rc;
    size_t size = 0;

    fd = open(path, O_RDWR);
    if (fd == -1) {
        perror(path);
```

```
        return -1;
    }

    rc = write(fd, "1", 2);
    if (rc <= 0) {
        perror("write");
        return -1;
    }

    do {
        rc = read(fd, buf, sizeof(buf));
        if (rc > 0)
            size += rc;
    } while (rc > 0);

    close(fd);
    return size;
}

static int scan_rom(char *path, char *file)
{
    struct dirent **namelist;
    char *name, *path2;
    int i, n, r, rc = 0, result = 0;
    struct stat buf;

    n = scandir(path, &namelist, 0, alphasort);
    if (n < 0) {
        perror("scandir");
        return -1;
    }

    for (i = 0; i < n; i++) {
        name = namelist[i]->d_name;

        if (fnmatch(".", name, 0) == 0)
            goto skip;
        if (fnmatch("..", name, 0) == 0)
            goto skip;

        path2 = malloc(strlen(path) + strlen(name) + 3);
        strcpy(path2, path);
        strcat(path2, "/");
        strcat(path2, name);

        if (fnmatch(file, name, 0) == 0) {
            rc = read_rom(path2);

            /*
             * It's OK if the ROM is unreadable.  Maybe there
             * is no ROM, or some other error ocurred.  The
             * important thing is that no MCA happened.
             */
            if (rc > 0)
                fprintf(stderr, "PASS: %s read %d bytes\n", path2, rc);
            else {
                fprintf(stderr, "PASS: %s not readable\n", path2);
                return rc;
            }
        } else {
            r = lstat(path2, &buf);
            if (r == 0 && S_ISDIR(buf.st_mode)) {
                rc = scan_rom(path2, file);
                if (rc < 0)
                    return rc;
            }
```

```
        }

        result |= rc;
        free(path2);

skip:
        free(namelist[i]);
    }
    free(namelist);
    return result;
}

int main(void)
{
    int rc;

    if (map_mem("/dev/mem", 0, 0xA0000, 1) == 0)
        fprintf(stderr, "PASS: /dev/mem 0x0-0xa0000 is readable\n");
    else
        fprintf(stderr, "FAIL: /dev/mem 0x0-0xa0000 not accessible\n");

    /*
     * It's not safe to blindly read the VGA frame buffer.  If you know
     * how to poke the card the right way, it should respond, but it's
     * not safe in general.  Many machines, e.g., Intel chipsets, cover
     * up a non-responding card by just returning -1, but others will
     * report the failure as a machine check.
     */
    if (map_mem("/dev/mem", 0xA0000, 0x20000, 0) == 0)
        fprintf(stderr, "PASS: /dev/mem 0xa0000-0xc0000 is mappable\n");
    else
        fprintf(stderr, "FAIL: /dev/mem 0xa0000-0xc0000 not accessible\n");

    if (map_mem("/dev/mem", 0xC0000, 0x40000, 1) == 0)
        fprintf(stderr, "PASS: /dev/mem 0xc0000-0x100000 is readable\n");
    else
        fprintf(stderr, "FAIL: /dev/mem 0xc0000-0x100000 not accessible\n");

    /*
     * Often you can map all the individual pieces above (0-0xA0000,
     * 0xA0000-0xC0000, and 0xC0000-0x100000), but can't map the whole
     * thing at once.  This is because the individual pieces use different
     * attributes, and there's no single attribute supported over the
     * whole region.
     */
    rc = map_mem("/dev/mem", 0, 1024*1024, 0);
    if (rc == 0)
        fprintf(stderr, "PASS: /dev/mem 0x0-0x100000 is mappable\n");
    else if (rc > 0)
        fprintf(stderr, "PASS: /dev/mem 0x0-0x100000 not mappable\n");
    else
        fprintf(stderr, "FAIL: /dev/mem 0x0-0x100000 not accessible\n");

    scan_tree("/sys/class/pci_bus", "legacy_mem", 0, 0xA0000, 1);
    scan_tree("/sys/class/pci_bus", "legacy_mem", 0xA0000, 0x20000, 0);
    scan_tree("/sys/class/pci_bus", "legacy_mem", 0xC0000, 0x40000, 1);
    scan_tree("/sys/class/pci_bus", "legacy_mem", 0, 1024*1024, 0);

    scan_rom("/sys/devices", "rom");

    scan_tree("/proc/bus/pci", "??.?", 0, 0xA0000, 1);
    scan_tree("/proc/bus/pci", "??.?", 0xA0000, 0x20000, 0);
    scan_tree("/proc/bus/pci", "??.?", 0xC0000, 0x40000, 1);
    scan_tree("/proc/bus/pci", "??.?", 0, 1024*1024, 0);

    return rc;
```

```
}
```