#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>

#include <fcntl.h>

#include <linux/fb.h>

#include <linux/kd.h>

#include <sys/mman.h>

#include <sys/ioctl.h>

#include <sys/time.h>

#include <string.h>

#include <errno.h>

struct fb\_var\_screeninfo vinfo;

struct fb\_fix\_screeninfo finfo;

char \*frameBuffer = 0;

//打印fb驱动中fix结构信息，注：在fb驱动加载后，fix结构不可被修改。

void

printFixedInfo ()

{

    printf ("Fixed screen info:\n"

            "\tid: %s\n"

            "\tsmem\_start: 0x%lx\n"

            "\tsmem\_len: %d\n"

            "\ttype: %d\n"

            "\ttype\_aux: %d\n"

            "\tvisual: %d\n"

            "\txpanstep: %d\n"

            "\typanstep: %d\n"

            "\tywrapstep: %d\n"

            "\tline\_length: %d\n"

            "\tmmio\_start: 0x%lx\n"

            "\tmmio\_len: %d\n"

            "\taccel: %d\n"

            "\n",

            finfo.id, finfo.smem\_start, finfo.smem\_len, finfo.type,

            finfo.type\_aux, finfo.visual, finfo.xpanstep, finfo.ypanstep,

            finfo.ywrapstep, finfo.line\_length, finfo.mmio\_start,

            finfo.mmio\_len, finfo.accel);

}

//打印fb驱动中var结构信息，注：fb驱动加载后，var结构可根据实际需要被重置

void

printVariableInfo ()

{

    printf ("Variable screen info:\n"

            "\txres: %d\n"

            "\tyres: %d\n"

            "\txres\_virtual: %d\n"

            "\tyres\_virtual: %d\n"

            "\tyoffset: %d\n"

            "\txoffset: %d\n"

            "\tbits\_per\_pixel: %d\n"

            "\tgrayscale: %d\n"

            "\tred: offset: %2d, length: %2d, msb\_right: %2d\n"

            "\tgreen: offset: %2d, length: %2d, msb\_right: %2d\n"

            "\tblue: offset: %2d, length: %2d, msb\_right: %2d\n"

            "\ttransp: offset: %2d, length: %2d, msb\_right: %2d\n"

            "\tnonstd: %d\n"

            "\tactivate: %d\n"

            "\theight: %d\n"

            "\twidth: %d\n"

            "\taccel\_flags: 0x%x\n"

            "\tpixclock: %d\n"

            "\tleft\_margin: %d\n"

            "\tright\_margin: %d\n"

            "\tupper\_margin: %d\n"

            "\tlower\_margin: %d\n"

            "\thsync\_len: %d\n"

            "\tvsync\_len: %d\n"

            "\tsync: %d\n"

            "\tvmode: %d\n"

            "\n",

            vinfo.xres, vinfo.yres, vinfo.xres\_virtual, vinfo.yres\_virtual,

            vinfo.xoffset, vinfo.yoffset, vinfo.bits\_per\_pixel,

            vinfo.grayscale, vinfo.red.offset, vinfo.red.length,

            vinfo.red.msb\_right, vinfo.green.offset, vinfo.green.length,

            vinfo.green.msb\_right, vinfo.blue.offset, vinfo.blue.length,

            vinfo.blue.msb\_right, vinfo.transp.offset, vinfo.transp.length,

            vinfo.transp.msb\_right, vinfo.nonstd, vinfo.activate,

            vinfo.height, vinfo.width, vinfo.accel\_flags, vinfo.pixclock,

            vinfo.left\_margin, vinfo.right\_margin, vinfo.upper\_margin,

            vinfo.lower\_margin, vinfo.hsync\_len, vinfo.vsync\_len,

            vinfo.sync, vinfo.vmode);

}

//画大小为width\*height的同色矩阵，8alpha+8reds+8greens+8blues

void

drawRect\_rgb32 (int x0, int y0, int width, int height, int color)

{

    const int bytesPerPixel = 4;

    const int stride = finfo.line\_length / bytesPerPixel;

    int \*dest = (int \*) (frameBuffer)

        + (y0 + vinfo.yoffset) \* stride + (x0 + vinfo.xoffset);

    int x, y;

    for (y = 0; y < height; ++y)

    {

        for (x = 0; x < width; ++x)

        {

            dest[x] = color;

        }

        dest += stride;

    }

}

//画大小为width\*height的同色矩阵，5reds+6greens+5blues

void

drawRect\_rgb16 (int x0, int y0, int width, int height, int color)

{

    const int bytesPerPixel = 2;

    const int stride = finfo.line\_length / bytesPerPixel;

    const int red = (color & 0xff0000) >> (16 + 3);

    const int green = (color & 0xff00) >> (8 + 2);

    const int blue = (color & 0xff) >> 3;

    const short color16 = blue | (green << 5) | (red << (5 + 6));

    short \*dest = (short \*) (frameBuffer)

        + (y0 + vinfo.yoffset) \* stride + (x0 + vinfo.xoffset);

    int x, y;

    for (y = 0; y < height; ++y)

    {

        for (x = 0; x < width; ++x)

        {

            dest[x] = color16;

        }

        dest += stride;

    }

}

//画大小为width\*height的同色矩阵，5reds+5greens+5blues

void

drawRect\_rgb15 (int x0, int y0, int width, int height, int color)

{

    const int bytesPerPixel = 2;

    const int stride = finfo.line\_length / bytesPerPixel;

    const int red = (color & 0xff0000) >> (16 + 3);

    const int green = (color & 0xff00) >> (8 + 3);

    const int blue = (color & 0xff) >> 3;

    const short color15 = blue | (green << 5) | (red << (5 + 5)) | 0x8000;

    short \*dest = (short \*) (frameBuffer)

        + (y0 + vinfo.yoffset) \* stride + (x0 + vinfo.xoffset);

    int x, y;

    for (y = 0; y < height; ++y)

    {

        for (x = 0; x < width; ++x)

        {

            dest[x] = color15;

        }

        dest += stride;

    }

}

void

drawRect (int x0, int y0, int width, int height, int color)

{

    switch (vinfo.bits\_per\_pixel)

    {

    case 32:

        drawRect\_rgb32 (x0, y0, width, height, color);

        break;

    case 16:

        drawRect\_rgb16 (x0, y0, width, height, color);

        break;

    case 15:

        drawRect\_rgb15 (x0, y0, width, height, color);

        break;

    default:

        printf ("Warning: drawRect() not implemented for color depth %i\n",

                vinfo.bits\_per\_pixel);

        break;

    }

}

#define PERFORMANCE\_RUN\_COUNT 5

void

performSpeedTest (void \*fb, int fbSize)

{

    int i, j, run;

    struct timeval startTime, endTime;

    unsigned long long results[PERFORMANCE\_RUN\_COUNT];

    unsigned long long average;

    unsigned int \*testImage;

    unsigned int randData[17] = {

        0x3A428472, 0x724B84D3, 0x26B898AB, 0x7D980E3C, 0x5345A084,

        0x6779B66B, 0x791EE4B4, 0x6E8EE3CC, 0x63AF504A, 0x18A21B33,

        0x0E26EB73, 0x022F708E, 0x1740F3B0, 0x7E2C699D, 0x0E8A570B,

        0x5F2C22FB, 0x6A742130

    };

    printf ("Frame Buffer Performance test...\n");

    for (run = 0; run < PERFORMANCE\_RUN\_COUNT; ++run)

    {

        /\* Generate test image with random(ish) data: \*/

        testImage = (unsigned int \*) malloc (fbSize);

        j = run;

        for (i = 0; i < (int) (fbSize / sizeof (int)); ++i)

        {

            testImage[i] = randData[j];

            j++;

            if (j >= 17)

                j = 0;

        }

        gettimeofday (&startTime, NULL);

        memcpy (fb, testImage, fbSize);

        gettimeofday (&endTime, NULL);

        long secsDiff = endTime.tv\_sec - startTime.tv\_sec;

        results[run] =

            secsDiff \* 1000000 + (endTime.tv\_usec - startTime.tv\_usec);

        free (testImage);

    }

    average = 0;

    for (i = 0; i < PERFORMANCE\_RUN\_COUNT; ++i)

        average += results[i];

    average = average / PERFORMANCE\_RUN\_COUNT;

    printf (" Average: %llu usecs\n", average);

    printf (" Bandwidth: %.03f MByte/Sec\n",

            (fbSize / 1048576.0) / ((double) average / 1000000.0));

    printf (" Max. FPS: %.03f fps\n\n",

            1000000.0 / (double) average);

    /\* Clear the framebuffer back to black again: \*/

    memset (fb, 0, fbSize);

}

int

main (int argc, char \*\*argv)

{

    const char \*devfile = "/dev/fb0";

    long int screensize = 0;

    int fbFd = 0;

    /\* Open the file for reading and writing \*/

    fbFd = open (devfile, O\_RDWR);

    if (fbFd == -1)

    {

        perror ("Error: cannot open framebuffer device");

        exit (1);

    }

    //获取finfo信息并显示

    if (ioctl (fbFd, FBIOGET\_FSCREENINFO, &finfo) == -1)

    {

        perror ("Error reading fixed information");

        exit (2);

    }

    printFixedInfo ();

    //获取vinfo信息并显示

    if (ioctl (fbFd, FBIOGET\_VSCREENINFO, &vinfo) == -1)

    {

        perror ("Error reading variable information");

        exit (3);

    }

    printVariableInfo ();

    /\* Figure out the size of the screen in bytes \*/

    screensize = finfo.smem\_len;

    /\* Map the device to memory \*/

    frameBuffer =

        (char \*) mmap (0, screensize, PROT\_READ | PROT\_WRITE, MAP\_SHARED,

                     fbFd, 0);

    if (frameBuffer == MAP\_FAILED)

    {

        perror ("Error: Failed to map framebuffer device to memory");

        exit (4);

    }

    //测试virt fb的性能

    performSpeedTest (frameBuffer, screensize);

    printf ("Will draw 3 rectangles on the screen,\n"

            "they should be colored red, green and blue (in that order).\n");

    drawRect (vinfo.xres / 8, vinfo.yres / 8,

             vinfo.xres / 4, vinfo.yres / 4, 0xffff0000);

    drawRect (vinfo.xres \* 3 / 8, vinfo.yres \* 3 / 8,

             vinfo.xres / 4, vinfo.yres / 4, 0xff00ff00);

    drawRect (vinfo.xres \* 5 / 8, vinfo.yres \* 5 / 8,

             vinfo.xres / 4, vinfo.yres / 4, 0xff0000ff);

    sleep (5);

    printf (" Done.\n");

    munmap (frameBuffer, screensize);    //解除内存映射，与mmap对应

    close (fbFd);

    return 0;

}