Raspberry Compote

```
#include <linux/kd.h>
#include <stdint.h>
#include "vcio.h"
#include <time.h>

// 'global' variables to store s
int fbfd = 0;
char *fbp = 0;
struct fb_var_screeninfo vinfo;
struct fb_fix_screeninfo finfo;
```

Pseudo-random ramblings about programming and other geeky stuff

Wednesday, 12 December 2012

Low-level Graphics on Raspberry Pi (part one)

The Raspberry Pi (RPi) comes built with hardware support - and supporting software programming libraries - for all the current state of the art standardised graphics goodies: OpenGL ES, OpenVG, EGL etc. and considering the performance gains of using the VideoCore GPU over the ARM CPU, it definitely makes sense to utilise these libraries to their full extent.

However, one of the main ideologies of the Raspberry Pi Foundation - the people who conceived the crafty little appliance we now know as RPi - was to introduce new generations to 'what goes behind the scenes' of fancy applications and user-interfaces. In my opinion, this goes as well for the 'fancy' graphics libraries and technologies. Therefore I would like to think it makes sense to introduce also the lower level interfaces for programming graphics on the RPi (most principles and some of the code I will introduce apply to other systems as well - *see comment 4 March 2016).

Basic command-line and file editing skills expected - some understanding of C programming would not hurt...

The lowest level graphics interface on a Linux system is the framebuffer (also see linux/fb.h). The framebuffer device - like most devices on a Linux system - can be opened as a file. The file can then be accessed for example using ioctl calls.

A basic example to open the framebuffer device and query the current display settings:

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <linux/fb.h>
#include <sys/mman.h>

// application entry point
int main(int argc, char* argv[])
{
  int fbfd = 0; // framebuffer filedescriptor
  struct fb_var_screeninfo var_info;

  // Open the framebuffer device file for reading and writing
  fbfd = open("/dev/fb0", O_RDWR);
  if (fbfd == -1) {
    printf("Error: cannot open framebuffer device.\n");
    return(1);
  }
  printf("The framebuffer device opened.\n");

  // Get variable screen information
  if (ioctl(fbfd, FBIOGET_VSCREENINFO, &var_info)) {
```

Blog Archive

- ► 2016 (6)
- ► 2015 (3)
- ► 2014 (9)
- ► 2013 (9)
- ▼ 2012 (2)
  - ▼ December (1)
    - Low-level Graphics on Raspberry Pi (part one)
  - ► November (1)

Code Repository

- Low-level Graphics on RPi

Discussion

- Low-level Graphics on RPi
- Python Programming on RPi
- Java Programming on RPi

Links

- Raspberry Pi
- Python

```
      printf("Error reading variable screen info.\n");
    }
    printf("Display info %dx%d, %d bpp\n",
           var_info.xres, var_info.yres,
           var_info.bits_per_pixel );

    // close file
    close(fbfd);

    return 0;

  }
```

Save (using your preferred text-editor) the above code to a file called **fbtest.c** (in your preferred directory/folder - I use a main directory called *projects* in the user's home directory and a couple of sub/directories... - then compile and link simply using the command (from command-line in the same directory the file is):

```
make fbtest
```

...this and it's output should look like:

```
pi@raspberrypi:~/projects/test/fbtest# make fbtest
cc fbtest.c -o fbtest
pi@raspberrypi:~/fbtest#
```

...and if you examine (list) the directory, you should see both the source file **fbtest.c** and the executable file **fbtest**:

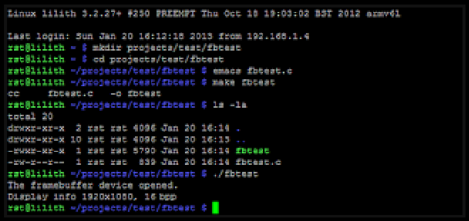
```
pi@raspberrypi:~/projects/test/fbtest# ls -la
total 20
drwxr-xr-x  2 rst rst 4096 Jan 20 16:09 .
drwxr-xr-x 10 rst rst 4096 Jan 20 16:05 ..
-rwxr-xr-x  1 rst rst 5790 Jan 20 16:09 fbtest
-rw-r--r--  1 rst rst  839 Jan 20 16:08 fbtest.c
```

Now you can run the executable using the command:

```
./fbtest
```

...you should see output similar to this (based on your display configuration - this is for a Full HD LCD connected through the HDMI on RPi and default configuration):

```
The framebuffer device opened.
Display info 1920x1080, 16 bpp
```



```
Linux lilith 3.2.27+ #250 PREEMPT Thu Oct 18 19:03:02 BST 2012 armv6l

Last login: Sun Jan 20 16:12:18 2013 from 192.168.1.4
rst@lilith ~ $ mkdir projects/test/fbtest
rst@lilith ~ $ cd projects/test/fbtest
rst@lilith ~/projects/test/fbtest $ emacs fbtest.c
rst@lilith ~/projects/test/fbtest $ make fbtest
cc     fbtest.c   -o fbtest
rst@lilith ~/projects/test/fbtest $ ls -la
total 20
drwxr-xr-x  2 rst rst 4096 Jan 20 16:14 .
drwxr-xr-x 10 rst rst 4096 Jan 20 16:13 ..
-rwxr-xr-x  1 rst rst 5790 Jan 20 16:14 fbtest
-rw-r--r--  1 rst rst  839 Jan 20 16:14 fbtest.c
rst@lilith ~/projects/test/fbtest $ ./fbtest
The framebuffer device opened.
Display info 1920x1080, 16 bpp
rst@lilith ~/projects/test/fbtest $
```

[Continues in part two]

Posted by Unknown at 16:56

Labels: C, graphics, Linux, Raspberry Pi

1 comment:

Raspberry Compote Friday, 4 March 2016 at 10:47:00 GMT

The support for the framebuffer driver and the functionality it provides seems to in fact differ a lot between Linux distributions and graphics hardware specific drivers. So unfortunately cannot promise the examples will fully work on even another Debian based system. See this post for some more information.

Reply

Enter your comment...

Comment as: Lhunden (Googl ▾ Sign out

Publish Preview ☐ Notify me

Note: only a member of this blog may post a comment.

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)