

Project 1

Graphics Library

Agenda

- Setting up
- Project Discussion

Setting up

- Make sure you can access `thoth.cs.pitt.edu`
- Set up manpages in thoth
- Download and install qemu in your local machine
- Usage of scp

Project 1 Discussion

Library Call	System Call(s) used
void init_graphics()	open, ioctl, mmap
void exit_graphics()	ioctl
char getkey()	select, read
void sleep_ms(long ms)	nanosleep
void clear_screen(void *img)	
void draw_pixel(void *img, int x, int y, color_t color)	
void draw_line(void *img, int x1, int y1, int x2, int y2, color_t c)	
void draw_circle(void *img, int x, int y, int r, color_t c)	
void *new_offscreen_buffer()	mmap
void blit(void *src)	

Project 1 Discussion

- Utilize man pages
- Submit 3 files:
 - library.c
 - graphics.h
 - a file where you write your driver code to test all the functions you wrote in library.c

init_graphics()

- Open the file `“/dev/fb0”` which represents the framebuffer
 - *open* system call
- Get the resolution and bit depth
 - Use *ioctl* system call

To get the screen size and bits per pixels, we can use a special system call: `ioctl`. This system call is used to `query and set parameters` for almost any `device` connected to the system. You pass it a file descriptor and a particular number that represents the request you're making of that device, plus a pointer to where the result will go (if applicable). We will use two requests: `FBIOGET_VSCREENINFO` and `FBIOGET_FSCREENINFO`. The first will give back (via the third parameter) a struct `fb_var_screeninfo` that will give the virtual resolution. The second will give back a struct `fb_fix_screeninfo` from which we can determine the bit depth. The total size of the `mmap()`'ed file would be the `yres_virtual` field of the first struct multiplied by the `line_length` field of the second.

init_graphics()

- Map the “/dev/fb0” file to memory using *mmap* system call
 - Instead of writing to file *fb0* using write() and seek() syscalls, you need to modify the bits of this mapped memory
 - Use MAP_SHARED parameter
- Change terminal settings using *ioctl* system call
 - Use TCGETS and TCSETS requests in ioctl
 - Unset ICANON bit and ECHO bit
 - Hint: Check the man page of *termios*
 - Canonical vs Non-Canonical
 - **DO NOT FORGET** to keep track of the old terminal settings

exit_graphics()

- **close** opened file
- **munmap** mapped memory
- reset to old terminal settings

getkey()

- Use select to detect if there is a character to be read
 - If so, read the input character and return it

sleep_ms()

- Sleep for the required amount of time
- Use the **nanosleep** system call
 - Check the man page on how to set the time of sleep

draw_pixel()

- Use pointer arithmetic to go the correct byte and draw a pixel there.
 - What does it mean to draw a pixel?
- Bit Depth: The number of bits in a pixel
 - In this case, it's 16 bits

draw_line() and draw_circle()

- Utilize the draw_pixel() function that you wrote

new_offscreen_buffer()

- Can draw here and then copy the bits to the actual framebuffer to display
- Just dynamically allocate memory space
 - Use **mmap**. Cannot use malloc() because it's not a system call
 - use MAP_PRIVATE | MAP_ANONYMOUS flag
 - set file descriptor as -1

blit()

- Memory Copy from offscreen buffer to the framebuffer
- Use loop(s) to change the bits

clear_screen()

- Loop over all the bytes of the image buffer parameter (which can be framebuffer or the offscreen_buffer) and set them to 0.

What's a color?

- Here, it's a 16-bit value
 - upper 5 bits = Red, middle 6 bits = Green, Last 5 bits = Blue
 - typedef the data type to color_t
- You can write a macro or a function
 - To combine separate R, G, B components to generate the 16-bit color value
 - Need to use bitwise operations for this