ReDS
Reconfigurable & embedded Digital Systems

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

# Architecture and Drivers for Smartphones

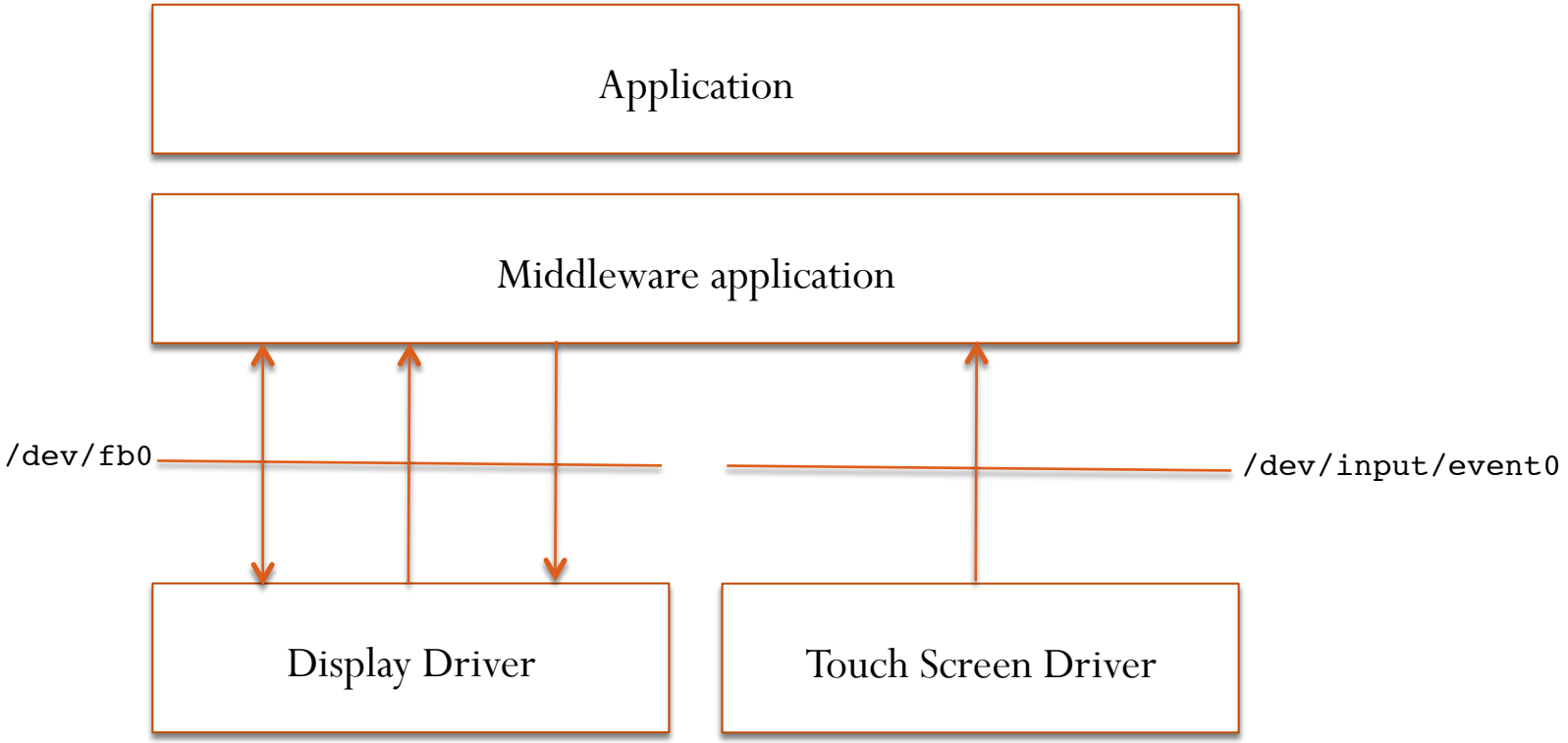
## *Introduction Labo 2*

Cours APS

Salvatore Valenza

Version 1.0 (2012-2013)

# Linux User Space vs Kernel Space



# ioctl() – input/output control

**SYNOPSIS**

```
#include <sys/ioctl.h>

int ioctl(int d, int request, ...);
```

**DESCRIPTION**

The ioctl function manipulates the underlying device parameters of special files. In particular, many operating characteristics of character special files (e.g. terminals) may be controlled with ioctl requests. The argument d must be an open file descriptor.

The second argument is a device-dependent request code. The third argument is an untyped pointer to memory. It's traditionally char *argp (from the days before void * was valid C), and will be so named for this discussion.

An ioctl request has encoded in it whether the argument is an in parameter or out parameter, and the size of the argument argp in bytes. Macros and defines used in specifying an ioctl request are located in the file <sys/ioctl.h>.

# mmap()

**SYNOPSYS**

```
#include <sys/mman.h>

void *mmap(void *addr, size_t len, int prot, int flags, int fildes, off_t off);
```

**DESCRIPTION**

The mmap() function shall establish a mapping between a process' address space and a file, shared memory object, or typed memory object. The format of the call is as follows:

```
    pa=mmap(addr, len, prot, flags, fildes, off);
```

The mmap() function shall establish a mapping between the address space of the process at an address pa for len bytes to the memory object represented by the file descriptor fildes at offset off for len bytes. The value of pa is an implementation-defined function of the parameter addr and the values of flags. The address range starting at pa and continuing for len bytes shall be legitimate for the possible (not necessarily current) address space of the process. The range of bytes starting at off and continuing for len bytes shall be legitimate for the possible (not necessarily current) offsets in the file, shared memory object, or typed memory object represented by files.

# Frame Buffer Data Structure – fb.h (1/3)

```
struct fb_fix_screeninfo {
        char id[16];                /* identification string eg "TT Builtin" */
        unsigned long smem_start; /* Start of frame buffer mem */
                                    /* (physical address) */
        __u32 smem_len;             /* Length of frame buffer mem */
        __u32 type;                 /* see FB_TYPE_* */
        __u32 type_aux;             /* Interleave for interleaved Planes */
        __u32 visual;               /* see FB_VISUAL_*        */
        __u16 xpanstep;             /* zero if no hardware panning */
        __u16 ypanstep;             /* zero if no hardware panning */
        __u16 ywrapstep;            /* zero if no hardware ywrap       */
        __u32 line_length;          /* length of a line in bytes       */
        unsigned long mmio_start; /* Start of Memory Mapped I/O      */
                                    /* (physical address) */
        __u32 mmio_len;             /* Length of Memory Mapped I/O */
        __u32 accel;                /* Indicate to driver which */
                                    /*  specific chip/card we have */
        __u16 reserved[3];          /* Reserved for future compatibility */
};
```

# Frame Buffer Data Structure – fb.h (2/3)

```
/*
 * Interpretation of offset for color fields: All offsets are from the right,
 * inside a "pixel" value, which is exactly 'bits_per_pixel' wide (means: you
 * can use the offset as right argument to <<). A pixel afterwards is a bit
 * stream and is written to video memory as that unmodified.
 *
 * For pseudocolor: offset and length should be the same for all color
 * components. Offset specifies the position of the least significant bit
 * of the pallette index in a pixel value. Length indicates the number
 * of available palette entries (i.e. # of entries = 1 << length).
 */
struct fb_bitfield {
        __u32 offset;                   /* beginning of bitfield  */
        __u32 length;                   /* length of bitfield
        */
        __u32 msb_right;

};
```

# Frame Buffer Data Structure – fb.h(3/3)

```
struct fb_var_screeninfo {
        __u32 xres;                /* visible resolution */
        __u32 yres;
        __u32 xres_virtual;        /* virtual resolution */
        __u32 yres_virtual;
        __u32 xoffset;             /* offset from virtual to visible */
        __u32 yoffset;             /* resolution */

        __u32 bits_per_pixel;      /* guess what */
        __u32 grayscale;           /* != 0 Graylevels instead of colors */

        struct fb_bitfield red;    /* bitfield in fb mem if true color, */
        struct fb_bitfield green; /* else only length is significant   */
        struct fb_bitfield blue;
        struct fb_bitfield transp;/* transparency */

        __u32 nonstd;              /* != 0 Non standard pixel format */

        __u32 activate;            /* see FB_ACTIVATE_*                */

        __u32 height;              /* height of picture in mm */
        __u32 width;               /* width of picture in mm  */
        ...
};
```

# Touch Screen – input.h

```
80 /*
81  * Event types
82  */
83
84 #define EV_SYN                  0x00
85 #define EV_KEY                  0x01
86 #define EV_REL                  0x02
87 #define EV_ABS                  0x03
88 #define EV_MSC                  0x04
89 #define EV_LED                  0x11
90 #define EV_SND                  0x12
91 #define EV_REP                  0x14
92 #define EV_FF                   0x15
93 #define EV_PWR                  0x16
94 #define EV_FF_STATUS            0x17
95 #define EV_MAX                  0x1f
```

```
516 /*
517  * Absolute axes
518  */
519
520 #define ABS_X                   0x00
521 #define ABS_Y                   0x01
522 #define ABS_Z                   0x02
523 #define ABS_RX                  0x03
524 #define ABS_RY                  0x04
525 #define ABS_RZ                  0x05
526 #define ABS_THROTTLE            0x06
527 #define ABS_RUDDER              0x07
528 #define ABS_WHEEL               0x08
529 #define ABS_GAS                 0x09
530 #define ABS_BRAKE               0x0a
531 #define ABS_HAT0X               0x10
532 #define ABS_HAT0Y               0x11
533 #define ABS_HAT1X               0x12
534 #define ABS_HAT1Y               0x13
535 #define ABS_HAT2X               0x14
536 #define ABS_HAT2Y               0x15
537 #define ABS_HAT3X               0x16
538 #define ABS_HAT3Y               0x17
539 #define ABS_PRESSURE            0x18
540 #define ABS_DISTANCE            0x19
541 #define ABS_TILT_X              0x1a
542 #define ABS_TILT_Y              0x1b
543 #define ABS_TOOL_WIDTH          0x1c
544 #define ABS_VOLUME              0x20
545 #define ABS_MISC                0x28
546 #define ABS_MAX                 0x3f
```