Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Odsek za računarsku tehniku i računarske komunikacije

Linuks u ugrađenim sistemima i razvoj rukovalaca

Radni okviri kernela za rukovaoce uređaja

Sadržaj

- Radni okviri kernela za rukovaoce uređaja
  - Koncept radnih okvira kernla
  - Ulazni podsistem

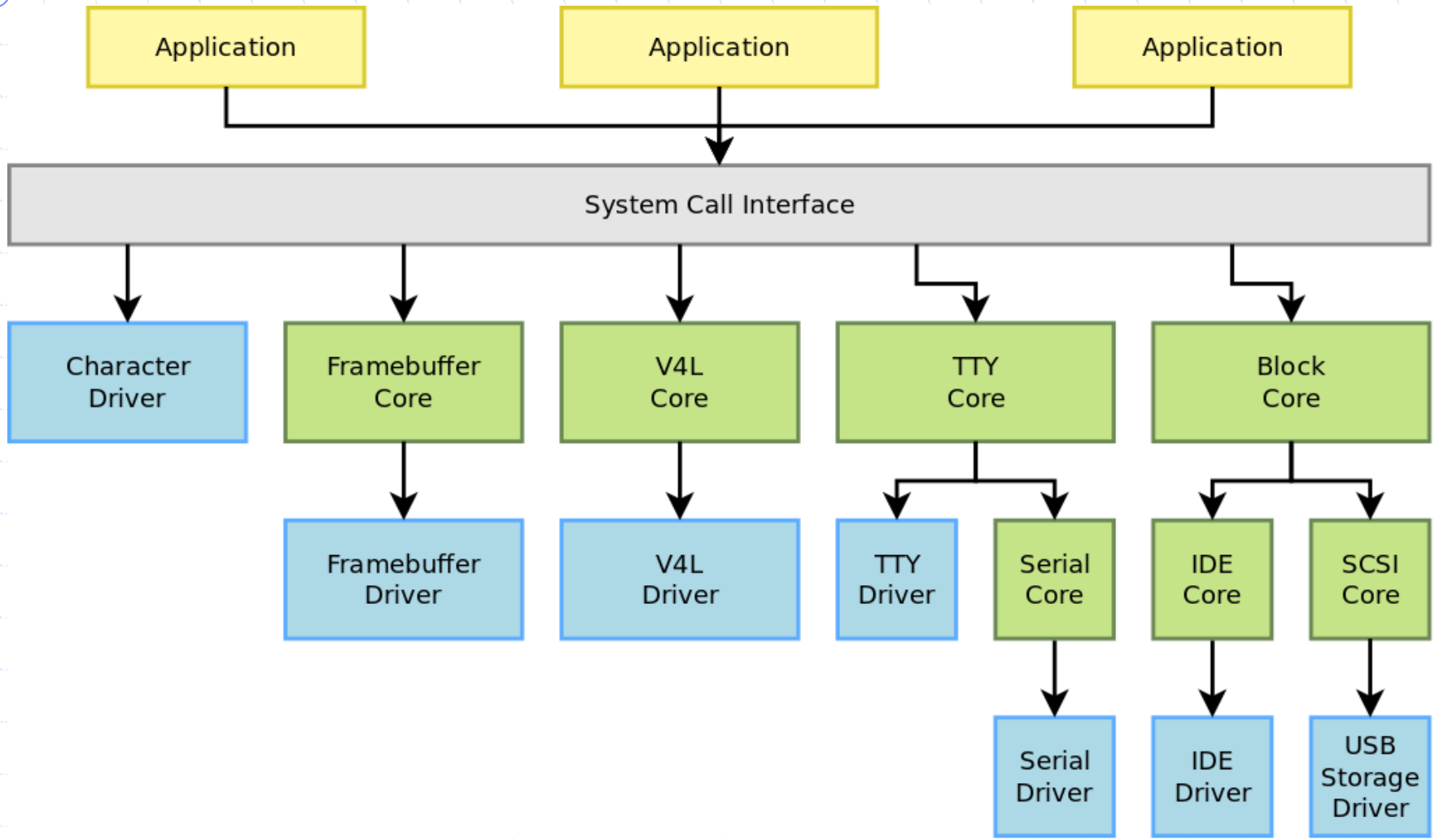
Koncept radnih okvira kernla

RADNI OKVIRI KERNELA ZA RUKOVAOCE UREĐAJA

Izvan slovnih rukovalaca: kernel frameworks

- Mnogi rukovaoci uređaja nisu implementirani kao karakterni rukovaoci
- Implementirani su pod radnim okvirom, specifičnim za dati tip uređaja (framebuffer, V4L, serial, itd)
  - Radni okvir dozvoljava razlaganje uobičajenih delova rukovalaca za isti tip uređaja
  - Iz korisničkog prostora, aplikacije ih i dalje vide kao karakterne uređaje
  - Radni okvir dozvoljava obezbeđivanje povezane sprege korisničkog prostora (ioctl i dr) za svaki tip uređaja, nebitno od korišćenog rukovaoca.

Radni okviri kernela



Primer: Framebuffer Radni okvir

- Opcija kernela CONFIG_FB
  - menuconfig FB
    - tristate "Support for frame buffer devices"
- Implementiran u C datotekama u drivers/video/fbdev/core
- Implementira jedinstven karakterni rukovalac i definiše user/kernel API
  - Prvi deo include/linux/fb.h
- Definiše set operacija koje framebuffer rukovalac mora da implementira i pomoćne funkcije za rukovaoce
  - struct fb_ops
  - Drugi deo include/linux/fb.h

Operacije Framebuffer rukovaoca

- Navedene su operacije koje framebuffer rukovalac može ili mora da implementira i da ih definiše u struct fb_ops strukturi

```
static struct fb_ops xxxfb_ops = {
        .owner = THIS_MODULE,
        .fb_open = xxxfb_open,
        .fb_read = xxxfb_read,
        .fb_write = xxxfb_write,
        .fb_release = xxxfb_release,
        .fb_check_var = xxxfb_check_var,
        .fb_set_par = xxxfb_set_par,
        .fb_setcolreg = xxxfb_setcolreg,
        .fb_blank = xxxfb_blank,
        .fb_pan_display = xxxfb_pan_display,
        .fb_fillrect = xxxfb_fillrect,   /* Potreban!!! */
        .fb_copyarea = xxxfb_copyarea,   /* Potreban!!! */
        .fb_imageblit = xxxfb_imageblit, /* Potreban!!! */
        .fb_cursor = xxxfb_cursor,       /* Opcioni !!! */
        .fb_rotate = xxxfb_rotate,
        .fb_sync = xxxfb_sync,
        .fb_ioctl = xxxfb_ioctl,
        .fb_mmap = xxxfb_mmap,
};
```

Kod Framebuffer rukovaoca

- U probe() funkciji, prijava framebuffer uređaja i operacija

```
static int xxxfb_probe (struct pci_dev *dev,
                        const struct pci_device_id *ent)
{
        struct fb_info *info;
        [...]
        info = framebuffer_alloc(sizeof(struct xxx_par), device);
        [...]
        info->fbops = &xxxfb_ops;
        [...]
        if (register_framebuffer(info) > 0)
                return -EINVAL;
        [...]
}
```

- register_framebuffer() kreira karakterni uređaj koji aplikacije korisničkog prostora mogu da koriste sa generičkim API-jem framebuffer-a

Struktura podataka specifična za rukovaoca

- Svaki radni okvir definiše strukturu koju rukovalac uređaja mora da prijavi da bi bio prepoznat kao uređaj u ovom radnom okviru
  - struct uart_port za serijske prolaze, struct netdev za mrežne uređaje, struct fb_info za framebuffer-e, itd.
- Dodatno, rukovalac obično mora da čuva dodatne informacije o svom uređaju
- Ovo se uobičajeno radi
  - Postavljanjem podklase odgovarajuće strukture radnog okvira
  - Čuvanjem reference na odgovarajuću strukturu radnog okvira
  - Ili uključivanjem naših informacija u strukturu radnog okvira

Primeri struktura podataka specifičnih za rukovaoce (1/2)

- i.MX serijski rukovalac: struct imx_port je podklasa struct uart_port

```
struct imx_port {
        struct uart_port port;
        struct timer_list timer;
        unsigned int old_status;
        int txirq, rxirq, rtsirq;
        unsigned int have_rtscts:1;
        [...]
};
```

- ds1305 RTC rukovalac: struct ds1305 sadrži referencu na struct rtc_device

```
struct ds1305 {
        struct spi_device *spi;
        struct rtc_device *rtc;
        [...]
};
```

Primeri struktura podataka specifičnih za rukovaoce (2/2)

- rtl8150 mrežni rukovalac: struct rtl8150 sadrži referencu na struct net_device i alociran je unutar strukture radnog okvira.

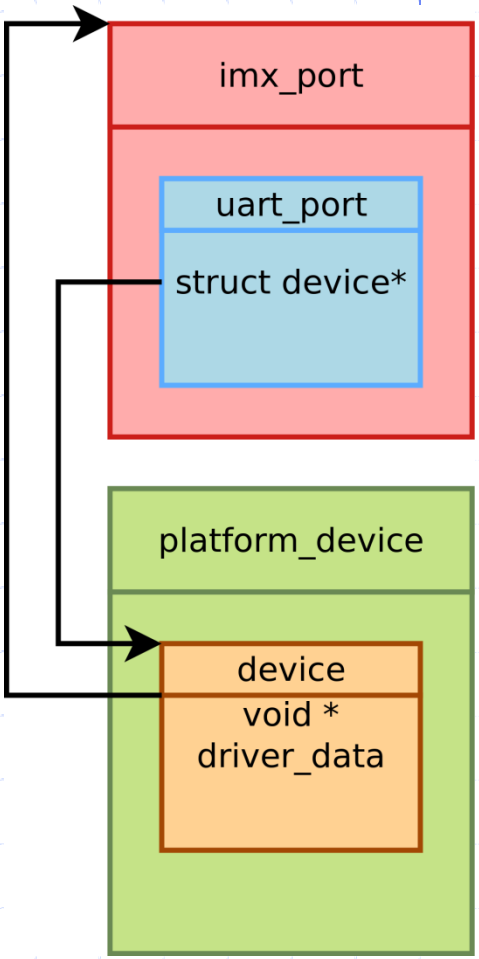
```
struct rtl8150 {
        unsigned long flags;
        struct usb_device *udev;
        struct tasklet_struct tl;
        struct net_device *netdev;
        [...]
};
```

Veza između struktura (1/4)

- Radni okvir obično sadrži struct device * pokazivač gde rukovalac mora da pokazuje na odgovarajući struct device
  - To je veza između logičkog uređaja (na primer mrežna sprega) i fizičkog uređaja (na primer USB mrežni adapter)
- Struktura uređaja takođe sadrži void * pokazivač koji rukovalac može slobodno da koristi.
  - Često se koristi da se uveže nazad uređaj ka strukturi višeg nivoa iz radnog okvira.
  - Na primer, to dozvoljava da se iz struct platform_device strukture pronađe struktura koja opisuje logički uređaj.

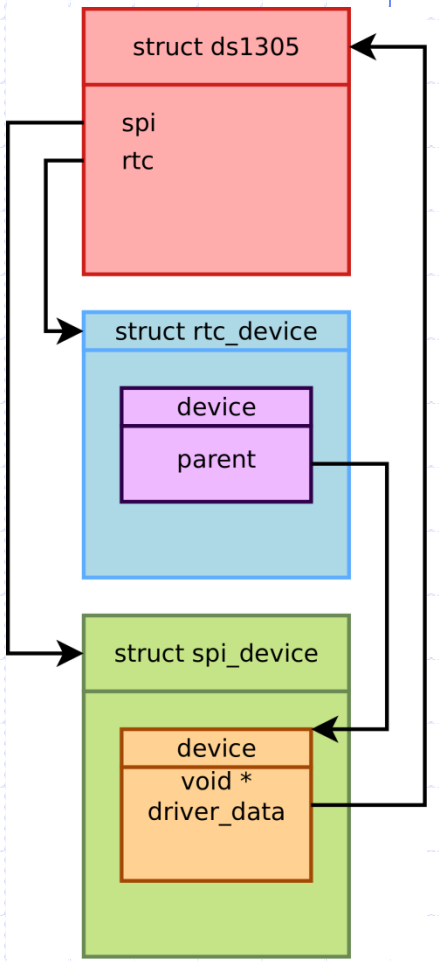
Veza između struktura (2/4)

```
static int serial_imx_probe(struct platform_device *pdev)
{
    struct imx_port *sport;
    [...]
    /* setup the link between uart_port and the struct
    * device inside the platform_device */
    sport->port.dev = &pdev->dev;
    [...]
    /* setup the link between the struct device inside
    * the platform device to the imx_port structure */
    platform_set_drvdata(pdev, sport);
    [...]
    uart_add_one_port(&imx_reg, &sport->port);
}
static int serial_imx_remove(struct platform_device *pdev)
{
    /* retrieve the imx_port from the platform_device */
    struct imx_port *sport = platform_get_drvdata(pdev);
    [...]
    uart_remove_one_port(&imx_reg, &sport->port);
    [...]
}
```



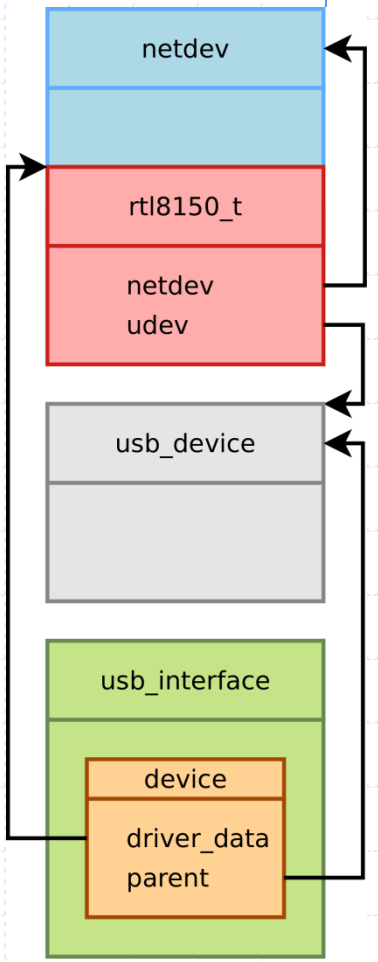
Veza između struktura (3/4)

```
static int ds1305_probe(struct spi_device *spi)
{
    struct ds1305 *ds1305;
    [...]
    /* set up driver data */
    ds1305 = devm_kzalloc(&spi->dev, sizeof(*ds1305),
                          GFP_KERNEL);
    if (!ds1305)
        return -ENOMEM;
    ds1305->spi = spi;
    spi_set_drvdata(spi, ds1305);
    [...]
    /* register RTC ... from here on, ds1305->ctrl needs
    *  locking */
    ds1305->rtc = devm_rtc_device_register(&spi->dev,
                  "ds1305", &ds1305_ops, THIS_MODULE);
    [...]
}
static int ds1305_remove(struct spi_device *spi)
{
    struct ds1305 *ds1305 = spi_get_drvdata(spi);
    [...]
}
```



Veza između struktura (4/4)

```
static int rtl8150_probe(struct usb_interface *intf,
                         const struct usb_device_id *id)
{
        struct usb_device *udev = interface_to_usbdev(intf);
        rtl8150_t *dev;
        struct net_device *netdev;
        netdev = alloc_etherdev(sizeof(rtl8150_t));
        dev = netdev_priv(netdev);
        [...]
        dev->udev = udev;
        dev->netdev = netdev;
        [...]
        usb_set_intfdata(intf, dev);
        SET_NETDEV_DEV(netdev, &intf->dev);
        [...]
}
static void rtl8150_disconnect(struct usb_interface *intf)
{
        rtl8150_t *dev = usb_get_intfdata(intf);
        [...]
}
```



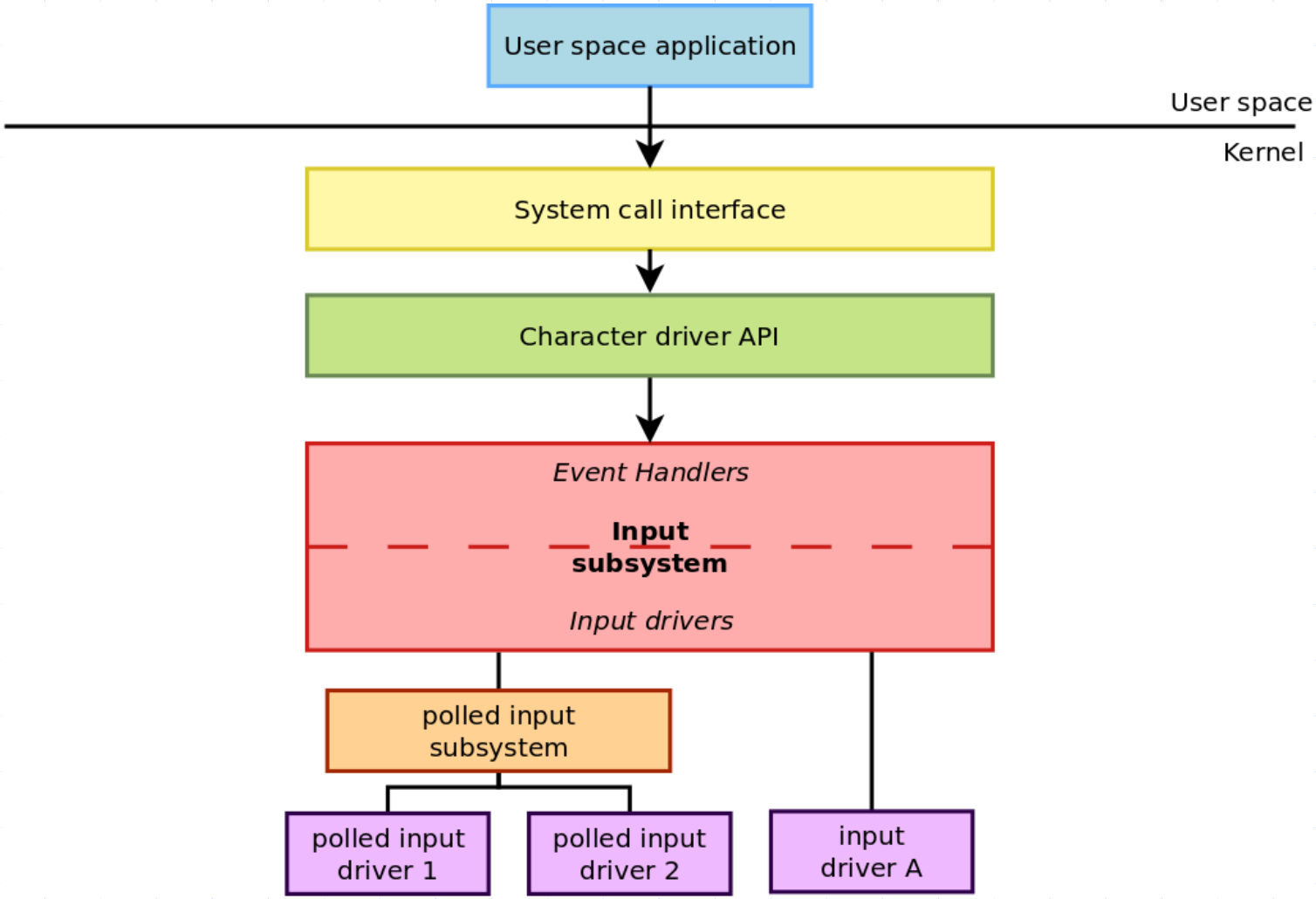
Ulazni podsistem

RADNI OKVIRI KERNELA ZA RUKOVAOCE UREĐAJA

Šta je ulazni podsistem?

- Ulazni podsistem se stara o svim ulaznim događajima koji dolaze od korisnika.
- Inicijalno je napisan za podršku USB HID (engl. Human Interface Device) uređaje, a brzo je preuzeo rukovanje svim vrstama ulaza (nezavisno od toga da li koriste USB): tastature, miševi, džojstici, ekrani osetljivi na dodir, itd.
- Ulazni podsistem je podeljen na dva dela:
  - Rukovaoci uređaja: komuniciraju sa fizičkom arhitekturom (na primer preko USB-a) i prosleđuju događaje (pomeraji miša, pritisci na tastere, koordinate dodira) ulaznom jezgru
  - Obrađivači događaja: prihvataju događaje od rukovalaca i prosleđuju ih gde su potrebni preko raznih sprega (najčešće preko evdev-a)
- U korisničkom prostoru najčešće se koriste od strane nekog grafičkog steka poput X.Org, Wayland ili Androidov InputManager.

Dijagram ulaznog podsistema



Pregled ulaznog podsitema (1/2)

- Opcija kernela CONFIG_INPUT
  - menuconfig INPUT
    - tristate "Generic input layer (needed for keyboard,mouse, ...)"
- Implementiran u drivers/input/
  - input.c, input-polldev.c, evbug.c

Pregled ulaznog podsitema (2/2)

- Implementira jedinstven karakterni rukovalac i definiše user/kernel API
  - include/uapi/linux/input.h
- Definiše set operacija koje ulazni rukovalac mora da implementira i pomoćne funkcije za rukovaoce
  - struct input_dev za deo rukovaoca uređajem
  - struct input_handler za deo obrađivača događaja
  - include/linux/input.h

API ulaznog podsistema (1/4)

- Ulazni uređaj je opisan sa veoma velikom struct input_dev strukturom, isečak:

```
struct input_dev {
        const char *name;
        [...]
        unsigned long evbit[BITS_TO_LONGS(EV_CNT)];
        unsigned long keybit[BITS_TO_LONGS(KEY_CNT)];
        [...]
        int (*getkeycode)(struct input_dev *dev,
                          struct input_keymap_entry *ke);
        [...]
        int (*open)(struct input_dev *dev);
        [...]
        int (*event)(struct input_dev *dev, unsigned int type,
                     unsigned int code, int value);
        [...]
};
```

- Pre nego što se koristi, ova struktura mora biti alocirana i inicijalizovana: struct input_dev *input_allocate_device(void);
- Nakon uklanjanja struct input_dev, mora biti oslobođena sa: void input_free_device(struct input_dev *dev);

API ulaznog podsistema (2/4)

- U zavisnosti od tipa događaja koji se generiše, ulazna bit polja evbit i keybit moraju biti podešena: na primer, za dugme se generiše samo EV_KEY tip događaja i od njih samo BTN_0 kod događaja:

```
set_bit(EV_KEY, myinput_dev.evbit);
set_bit(BTN_0, myinput_dev.keybit);
```

- set_bit() je atomska operacija koja postavlja određeni bit na 1 (kasnije objašnjeno).
- Kada se ulazni uređaj alocira i popuni, poziva se funkcija za prijavljivanje:
  int input_register_device(struct input_dev *);
- Kada se ukolni rukovalac, ulazni uređaj se uklanja korišćenjem:
  void input_unregister_device(struct input_dev *);

API ulaznog podsistema (3/4)

- Događaji se šalju od rukovaoca ka obrađivaču događaja korišćenjem input_event(struct input_dev *dev, unsigned int type, unsigned int code, int value);
- Tipovi događaja su dokumentovani u: Documentation/input/event-codes.txt

API ulaznog podsistema (4/4)

- Događaj se sastoji od jedne ili više ulaznih promena podataka (paketi ulaznih promena podataka) poput stanja dugmeta, relativne ili apsolute pozicije na nekoj osi, itd.
- Nakon slanja potencijalno višestrukih događaja, ulazno jezgro mora biti notifikovano pozivom funkcije: void input_sync(struct input_dev *dev):
- Ulazni podsistem obezbeđuje omotače (wrappers) kao što su input_report_key(), input_report_abs(), ...

Polovana ulazna podklasa

- Ulazni podsistem obezbeđuje podklasu koja podržava jednostavne ulazne uređaje koji ne podižu prekide, već moraju da se periodično skeniraju ili poluju da bi se primetile izmene u njihovom stanju.
- Polovani ulazni uređaj je opisan strukturom struct input_polled_dev:

```
struct input_polled_dev {
        void *private;
        void (*open)(struct input_polled_dev *dev);
        void (*close)(struct input_polled_dev *dev);
        void (*poll)(struct input_polled_dev *dev);
        unsigned int poll_interval; /* msec */
        unsigned int poll_interval_max; /* msec */
        unsigned int poll_interval_min; /* msec */
        struct input_dev *input;
/* private: */
        struct delayed_work work;
}
```

API polovanog ulaznog podsistema

- Alokacija/oslobađanje strukture struct input_polled_dev se radi korišćenjem:
  - input_allocate_polled_device()
  - input_free_polled_device()
- Samo poll() metoda je obavezna za strukturu struct input_polled_dev, ova funkcija poluje uređaj i šalje ulazne događaje.
- Polja id, name, evkey i keybit od ulaznih polja moraju biti inicijalizovana.
- Ako nijedno od poll_interval polja nije popunjeno, onda je podrazumevani poll_interval 500ms.
- Prijavljivanje/uklanjanje uređaja se radi sa:
  - input_register_polled_device(struct input_polled_dev *dev).
  - input_unregister_polled_device(struct input_polled_ dev *dev)

evdev sprega korisničkog prostora

- Glavna sprega korisničkog prostora za ulazne uređaje je sprega događaja
- Svaki ulazni uređaj se predstavlja sa /dev/input/event<X> karakternim uređajem
- Aplikacija korisničkog prostora može da koristi blokirajuća i neblokirajuća čitanja, ali takođe i select() (da bude obaveštena o događajima) nakon otvaranja uređaja.
- Svako čitanje vraća strukturu struct input_event sledećeg formata:

```
struct input_event {
        struct timeval time;
        unsigned short type;
        unsigned short code;
        unsigned int value;
};
```

- Veoma korisna aplikacija za testiranje ulaznih uređaja je evtest, sa http://cgit.freedesktop.org/evtest/