# 21. Tétel

Ismertesse a karakteres eszközvezérlő implementációját a Linux rendszerben!

## Eszközkezelők

* állományok Linuxban (/dev/)
  + egyszerű állomány
  + könyvtár
  + eszközök
    - blokk
    - karakter
  + szimbolikus link
* eszközvezérlő implementálása: függvények készítése …
  + állománnyal való műveletekre
    - megnyitás
    - írás
    - olvasás
    - bezárás
  + modulkezelésre
    - init
    - tisztogató
  + kernel nyilvántartásába vételére
  + kernel nyilvántartásából eltávolításra

## Egy eszköz azonosítója

* major
  + azonosítja az állományhoz/eszközhöz tartozó drivert
  + fő eszközvezérlő csoport
    - ttyXX – 4 – char
    - hdXX – 3 – block
    - sdXX – 8 - block
    - kbdXX – 11 – char – keyboard
  + kernel elintézi az ezen a szinten történő dolgokat
* minor
  + meghatározza egész pontosan melyik eszközre hivatkozunk
  + lehetővé teszi, hogy több azonos típusú eszközt csatoljunk
    - a meghajtó használhat eltérő implementációt a különböző minorokhoz, de az egységes viselkedés ajánlott
  + felhasználói programok kezelik
  + eszköz azonosítása
    - ttyS0 – 64
    - hda – 0
    - sda – 0

## Eszköz regisztrálása, eltávolítása

* **Regisztrálás**
  + virtuális állománykezelő függvényeinek fő azonosítóhoz rendelése
  + int register\_chrdev(unsigned int major, const char \*name, struct file\_operations \*fops);
    - negatív visszatérési érték – ERROR
    - major – fő azonosító
      * rossz meghajtó választásának kockázata
      * 0 érték – dinamikusan a rendszer is meg tudja határozni (visszatérési értékben megadja)
    - fops – strktúrára mutató – az állománykezelési funkciókat megvalósító függvények mutatóit tartalmazza
* **Modul eltávolítása**
  + void unregister\_chrdev(unsigned int major, const char \*name);
    - ezsközvezérlő neve: biztonsági okokból 🡪 a kernel összehasonlítja a korábban regisztrálttal, és ltérés esetén nem hajtja végre

## Eszközállomány dinamikus létrehozása, eltávolítása

* **állományok nyilvántartása**
  + eredetileg minden Linux által támogatott eszközhöz volt egy dummy állomány a /dev/-ben
  + aztán virtuális devfs használata
  + ma: udev
* **udev**
  + felhasználói módban futó alkalmazás (udev daemon)
  + fogadja a kernel hotplug üzeneteit 🡪 dinamikusan töröl , vagy létrehoz eszközállományokat a memóriában tárolt virtuális eszköz-állományrendszerben
  + hotplug üzenetek
    - Linux eszközmodell
      * Tartalmazza:
        + eszköz típus osztály
        + melyik buszra csatlakozik
        + speciális attribútumok
    - be kell regisztrálni az eszközhöz tartozó eszközmodellt
* **eszközállomány automatikus létrehozása:**
  + struct device\* device\_create(struct class \*osztály, struct device \*szülő, dev\_t eszköz, const char \*név, …);
    - osztály – eszköz-osztály
      * létrehozása: struct class \*class\_create(struct module\* modul, const char \*név);
        + modul – a modul makrója, ami az eszközállományhoz tartozik (THIS\_MODULE makróval lehet átadni a jelenlegit)
        + név – az osztály szöveges megnevezése
      * megszüntetés: void class\_destroy(struct class \*osztály);
    - szülő – szülőeszköz, ha ilyen létezik
    - eszköz – major, minor azonosítókat tartalmazó argumentum
      * előállítható: MKDEV() makróval
* **eszköz eltávolítása:** void device\_destroy(struct class\* osztály, dev\_t eszköz);

## Állományműveletek implementálása

* file\_operations struktúra: függvények mutatóinak struktúrája
  + mit történjen megnyitáskor, olvasáskor, íráskor stb ..
  + NULL pointer: az adott eszköz nem támogatja az adott műveletet
  + a default leírását a linux/fs.h állomány tartalmazza (/usr/include/ or /usr/locale/include/)
  + elemei (többek között)
    - *owner: THIS MODULE*
    - *read*
    - *write*
    - *open*
    - *release*

## Adatmozgatás Kernel és User space között

* #include <linux/uaccess.h>
* read/write függvények bufferei user-space-ben vannak ⬄ kernel modulok által foglalt területek: kernel space-ben 🡪 a kettő nem lát rá egymásra 🡪 meg kell oldani az átmozgatást
* előre definiált függvények a problémára
  + copy\_from\_user
  + copy\_to\_user

## Minor azonosító használata

* pl. többszörös soros kommunikáció
* egy driver kell, de több állomány használata célszerű
* a mellékazonosító megválasztása ránk van bízva
* guideline:
  + olvasást/írást kezelő függvényekben helyezzük az elágazásokat, amelyek mellékazonosító alapján másképp fognak viselkedni
  + állomány megnyitásakor eltérő lekezelő függvényeket állítunk be (paraméter átadás?)