Linux Character Drivers (SW)

Hvad er en character driver?

- En driver hvis data repræsenteres som en **strøm af karakterer**.
- Read og write operationer til forskellige steder i filen er unødvendig.
- Bliver eksempelvis brugt til **seriel kommunikation.**
- Char drivers bliver tilgået som **node** i filsystemet. -> bliver gemt i /dev/-mappen i Linux.

Forklar hvad major-/minor numre er og hvordan de kan allokeres?

- **Major nummer & minor nummer:**
    - Angiver hvilket modul/driver der bliver refereret til
    - Typisk **et major nummer per driver**
    - Bliver brugt til at **genkende** moduler i systemet
    - Kernen bruger majornummeret til at **linke** filen til driveren/modulet
    - En driver med et major nummer kan **tilgås af forskellige noder,** som får deres eget minor nummer
    - **Allokering:**
        - Major nummer kan allokeres på **2 måder** ved indsættelse af modul**:**
            - **Statisk:**
                - Bruges ved **små embedded systemer** hvor man har kontrol over alle moduler og majornumre
                - **register_chrdev_region(dev_t,unsigned int, char*):**
                    - parameter 1: Majornummeret på driveren
                    - parameter 2: Antal devices (minors)
                    - parameter 3: Navnet på device der forbindes med majornummeret
            - **Dynamisk:**
                - Generelle **større systemer** (PC'er) bruger dynamisk allokering
                - Typisk så kender man ikke majornummeret i forvejen
                - **Majornummeret oprettes dynamisk**
                - **alloc_chrdev_region(dev_t*,unsigned int,char*):**
                    - parameter 1: Her bliver majornummeret lagt i
                    - parameter 2: minornummeret
                    - parameter 3: Navnet på device

Hvordan registrerer man et device og hvad sker der når man gør det?

- **cdev_add(struct cdev*, dev_t, unsigned int):**
    - parameter 1: cdev strukturen for devicen
    - parameter 2: minornummeret
    - parameter 3: navnet på devicen
    - Herefter er driveren **kørende** -> man kan åbne, skriver, læse og lukke den

Forklar formålet med metoderne open/close/read/write?

- **open**:
    - **Initiere** og **klargøre** en driver for senere operationer.
    - Den tjekker for **fejl i device**
    - **Allokere** hukommelse til brug
- **close/release:**
    - Deallokerer alt som open har allokeret og lukker devicen
- **read:**
    - Denne funktion bliver kaldt når der vil læses fra driver noden fra applikationslageret
- **write:**
    - Denne funktion bliver kaldt når der vil skrives til driver noden fra applikationslageret

Forklar hvordan data overføres mellem user-/kernel space?

- Kernel og user space har hver deres **egen hukommelses lokatione**r.
- De har i**kke adgang** til hinandens -> Det giver en **page fault**
- -> **copy_to_user** & **copy_from_user**