Eksamen 2012  
3. semester  
I3MPS - Microprocessor systemer  
Kasper Nissen

**Spørgsmål 4**

**Linux Character Drivers**

**Spørgsmål**

* Hvad er en character driver?
* Forklar hvad major-/minor numre er og hvordan de kan allokeres
* Hvordan registerer man et device og hvad sker der når man gør det?
* Forklar formålet med metoderne Open/Close/Read/Write
* Forklar hvordan data overføres mellem user-/kernel space.

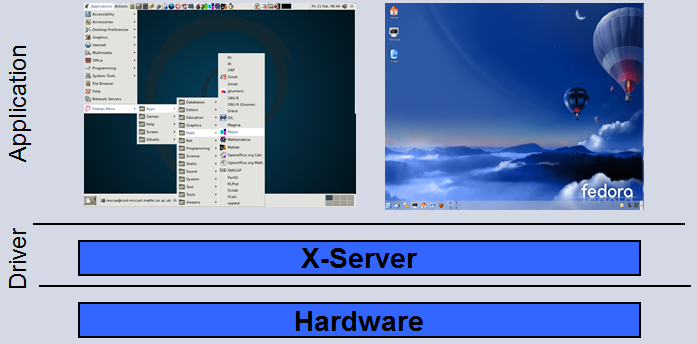
**Pensum:**

* Linux Devise Driver 1-70

**Exercise 4 & 5**

**Linux Device Drivers generelt**

Device drivere er en slags ”black box” som får et specielt hardware element til at fungere sammen med et program interface. Detaljerne om hvordan device driveren virker er skjult. Men ved en række standardiseret kald som er individuelle for en specifik driver, kan man altså få hardware enheden til at udføre det man gerne vil. En device driver opbygges som et kernemodul, som så indsættes i kernen når driveren ønskes brugt.



**Kernemoduler:**

Linux device drivere implementeres ofte til kernen, der findes dog også drivere som implementeres til user space, kaldet user space drivere. Kernemodulet ændret kernens funktion i run-time. Kernemoduler adskiller sig fra alm. Programmer:

* Det er object kode, som ikke linkes til en executable
* Bliver linket til kernen ved at bruge programmet insmod
* Kan kun bruge den funktionalitet som er tilstede i kernen
* Har ikke adgang til standard c/c++ biblioteker
* Man kan sige at et modul minder om et bibliotek
* Det kan være fatalt hvis der sker fejl i et kernemodul, da der ikke er nogen sikkerhed i kernen.

Metoder til at indsætte og udtage et modul:

* Insmod: Bruges til at indsætte et kernemodul i kernen.
* Rmmod: Bruges til at udtage et kernemodul
* Lsmod: Bruges til at frembringe en liste over indsætte kernemoduller
* Modprobe: Fungere på samme måde som insmod, kan bruges hvis man har flere moduller i en mappe der skal indsættes

Fordele ved user space drivere frem for kernels pace

* Langt mere beskyttelse
* Brug af C/C++ standard biblioteker
* En proces kan nemt stoppes (kill pid)

Ulemper ved user space drivere frem for kernels pace

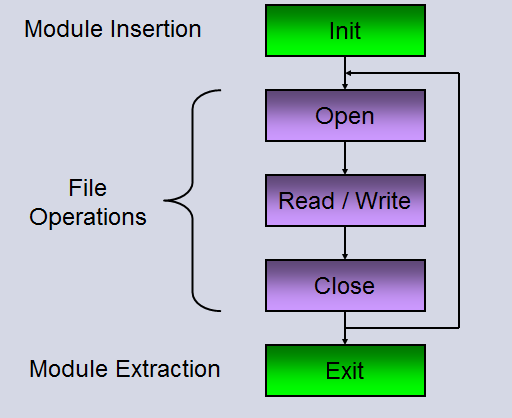
* Der kan ikke bruges interrupts
* Ingen direkte adgang til memory
* Somme tider langsommere

**Linux Character Drivers**

Et character device eller char device, er et device der kan tilgåes som en strøm af bytes(lidt ligesom en fil). En char driver bruges til at implementere denne adfærd. Normalt vil en sådan driver minimum indeholde system kaldene:

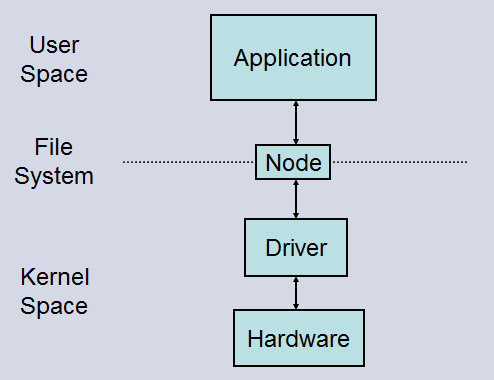
* Open
  + Initiering og klargøring af en driver for senere operationer
  + Tjekker for fejl i devicet
  + Allokere hukommelse til brug
* Close/release
  + Deallokering af alt som open har allokeret og lukker device.
* Read
  + Denne funktion bliver kaldt når der vil læses fra driver noden fra applikationslageret.
* Write
  + Denne funktion bliver kaldt når der vil læses fra driver noden fra applikationslageret.

Et eksempel på dette er serial porte som i linux ligger i /dev/ttyS0 mfl. Da de alle sammen repræsentere en strøm af bytes. Måden man tilgår disse devices på er igennem filsystemts noder, som f.eks. /dev/tty1. Forskellen på en char device og en fil, er at man i en fil altid vil kunne gå frem og tilbage, modsat char devicet som oftest blot er en data kanal, som kun kan tilgåes sekventielt. Der findes dog nogle char devices som ligner data områder, som kan gå frem og tilbage.



**Major**

* Et nummer pr driver
* Allokeres ved indsættelse af kernemodulet
  + Dynamisk
  + Statistik
* Angiver hvilket modul/driver der bliver refereret til
* 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 tilgåes af forskellige noder, som får der eget minor nummer.

**Minor**

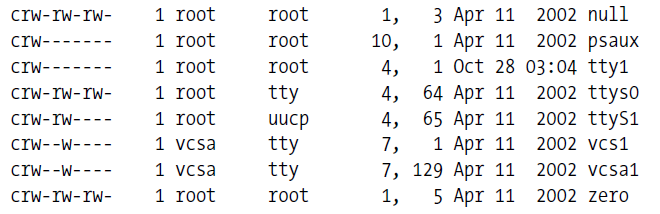
* Et nummer pr. device
* Et specifikt device.

Char devices tilgåes via navne i filsystemet. Noder bliver allokeret i mappen /dev. Når der skal oprettes en node bruges følgende kommando:

mknod /dev/NAVN PÅ MODUL c MAJOR MINOR

c – angiver at det er en char driver der indsættes.

Ved at bruge kommandoen ls –l fås en liste over kørende drivere og deres major/minor numre. Et eksempel på dette ses nedenfor:



Traditionelt angiver major nummeret driveren som er associeret med devicet. Det er dog ikke altid det bliver overholdt længere. Som det ses ovenfor så bliver driveren /dev/null og /dev/zero begge styret af driver 1, hvor vcs1 f.eks. bliver styret af driver 7.

**Allokering**

Minor nummeret bruges af kernen til at bestemme præcis hvilket device der refereres til. Der er 2 måder at tildele disse numre på.

Statisk

Det kan enten gøres manuelt ved følgende i init delen af kernemodullet:



* Parameter 1: Majornummeret på driveren
* Parameter2: Antal devices(minors)
* Parameter3: Navnet på device der forbindes med majornummeret.

Ulempen ved manuelt at gøre det er at man skal være sikker på man ikke tager et device nummer som allerede er i brug. Bruges ved **små embedded systemer**, hvor man har kontrol over alle moduler og majornumre.

Dynamisk

Derfor forsøger mange i Linux kernel development community at gå over til dynamisk at allokere device numre, ved følgende kommando:



* Parameter 1: Her bliver majornummeret lagt i
* Parameter 2: Første minor
* Parameter3: Antal af minornummer
* Parameter4: Navnet på device

Bruges på større systemer (PC’er). Typisk vil man ikke kende majornummeret i forvejen.

Deallokering

Man skal dog huske at frigøre disse numre på igen når modullet tages ud af kernen. Og dette gøres i exit funktionen med følgende:



* Parameter1: Majornummer
* Parameter2: antal minors

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

Et device registreres med følgende kommando:

Cdev\_add(struct cdev\*, dev\_t, unsigned int)

* Parameter1: cdev strukturen for device
* Parameter2: minornummeret
* Parameter3: Navnet på device

Herefter er driveren kørende -> Man kan altså nu åbne, skrive, læse og lukke den.

**Dataoverførsel mellem user-/kernels pace?**

Som sagt så er måden man tilgår devices på ved brug af noder, som oprettes ved brug af mknod.

Når noden er oprettet og har fået en fil i mappen /dev. Kan de forskellige metoder tilgåes fra userspace, ved f.eks. at bruge cat til at læse en værdi, eller echo for at skrive en værdi til noden og dermed til kernemodullet.

Dog skal man huske på følgende:

* Kernel og user space har hver deres egen hukommelses lokationer
* De har ikke adgang til hinandens hukommelse -> Det vil give en page fault.
* Følgende funktioner kan bruges:
  + Copy\_to\_user
  + Copy\_from\_user