IOCTL + Kernel Timers

- *Hvad benytter man IOCTL kald til?*
- *Hvordan implementeres IOCTL funktioner i en device driver og hvordan tilgås disse fra user-space?*
- *Hvordan benytter man timers / delays i kernen?*
- *Hvilke fordele / ulemper har de forskellige delay / timer typer?*

Hvad benytter man IOCTL kald til?

- Direkte systemkald
- Tilgå modul fra userspace
- Opsætning af modul fra userspace
- IOCTL kald er globale i kernen
- 'Input/Output control'

Hvordan implementeres IOCTL funtioner i en device driver og hvordan tilgås disse fra user-space?

- Først skal man finde et ledigt IOCTL nummer(Slåes op i ioctl-number.txt)
- Der oprettes en IOCTL metode

```
long my_ioctl(struct file *fp, unsigned int cmd, unsigned long arg)
```

- Metoden registreres i fops structen

```
.unlocked_ioctl = my_ioctl
```

- IOCTL kommandoer oprettes, og kan defineres som macroer:

```
#define START _IO(MY_IOCTL_NO, 1)
......
```

- IOCTL metoden implementeres til at kunne gøre forskellige ting på IOCTL kommandoerne
  - Switch/case der kalder forskellige metoder, sætter variabler osv.
- IOCTL kald kan nu benyttes fra userspace:
  - Userspace applikationen skal kende til IOCTL nummeret
  - En file descriptor åbnes til device node

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

Hvordan benytter man timers / delays i kernen?

- Den relative tid kan fås fra jiffies i Linux

  - jiffies er en 64 bit værdi
  - Genstartes ved system start
  - 'flips' efter 2^{64}

- **Busy waiting**
  - En 'tom' while løkke
- **Wait interruptible, wait interruptible timeout**
- **Scheduler timeout**
- **Sleep**
- Kernel timere er baseret på software interrupts
- Timer funktioner køres i interrupt context
  - Ingen sleeps, allokering, osv..

Hvilke fordele / ulemper har de forskellige delay / timer typer?

- **Busy waiting:**
  - Holder på sin CPU tid, altså processen bliver ikke flyttet ud af schedulerens wait-queue
  - Bruger mange CPU resourcer
- **Wait interruptible**
  - Vånger når der er brug for den
  - Sleeping
- **Kernel Timers**
  - Opfører sig som et planlagt interrupt
  - Skal følge samme regler som en ISR
  - Ikke i process kontekst