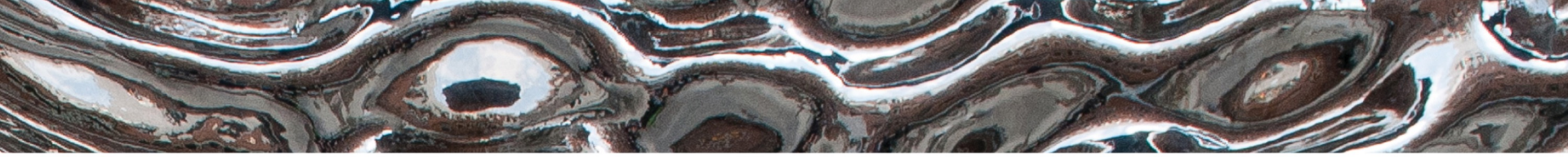


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

**IN1020 - Introduksjon til datateknologi**

Forelesning – 08.09.2022

# *Operativsystemer – En liten introduksjon*

*Håkon Kvale Stensland & Carsten Griwodz*

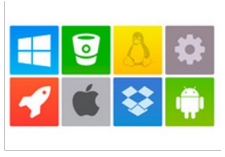

simula

tfs

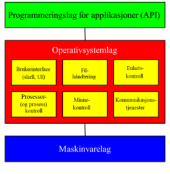
# Plan for ”nettverksdelen” av IN1020

- **8. september - Introduksjon til operativsystemer**
- 20. oktober – Datanett – En liten introduksjon
- 26. oktober – Historie, mer om lagdeling og protokoller
- 27. oktober – Kryptering i datakommunikasjon og som sikkerhetstiltak
- 2. november – Hvordan fungerer din trådløse ruter?
- 3. november – Tjenester i Internett

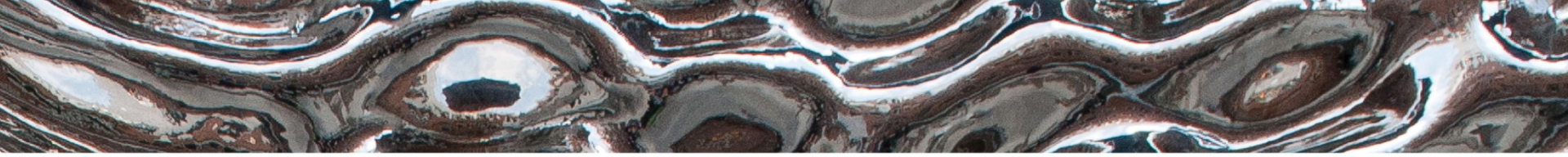
# Forelesningen i dag skal gi dere en kort introduksjon til datateknologi og hvordan en datamaskin fungerer.



Hvordan fungerer datamaskinen?




Operativsystemer: Hvordan fungerer de?



UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

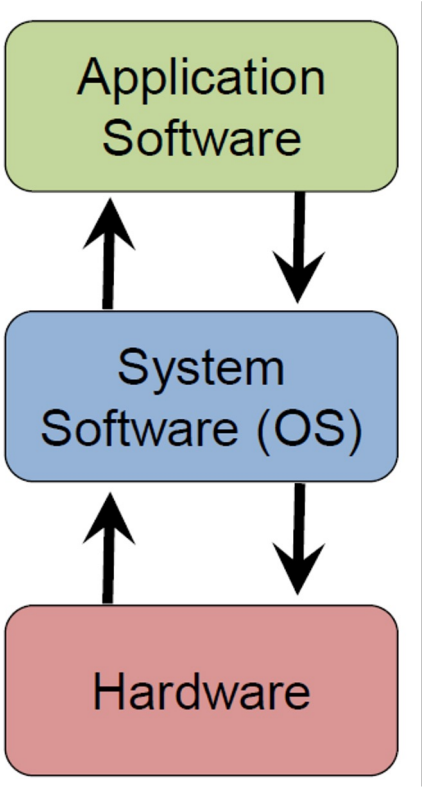
**Hvordan fungerer en datamaskin?**

Alle datamaskiner og «duppedingser» trenger programvare for å gjøre noe som helst fornuftig…

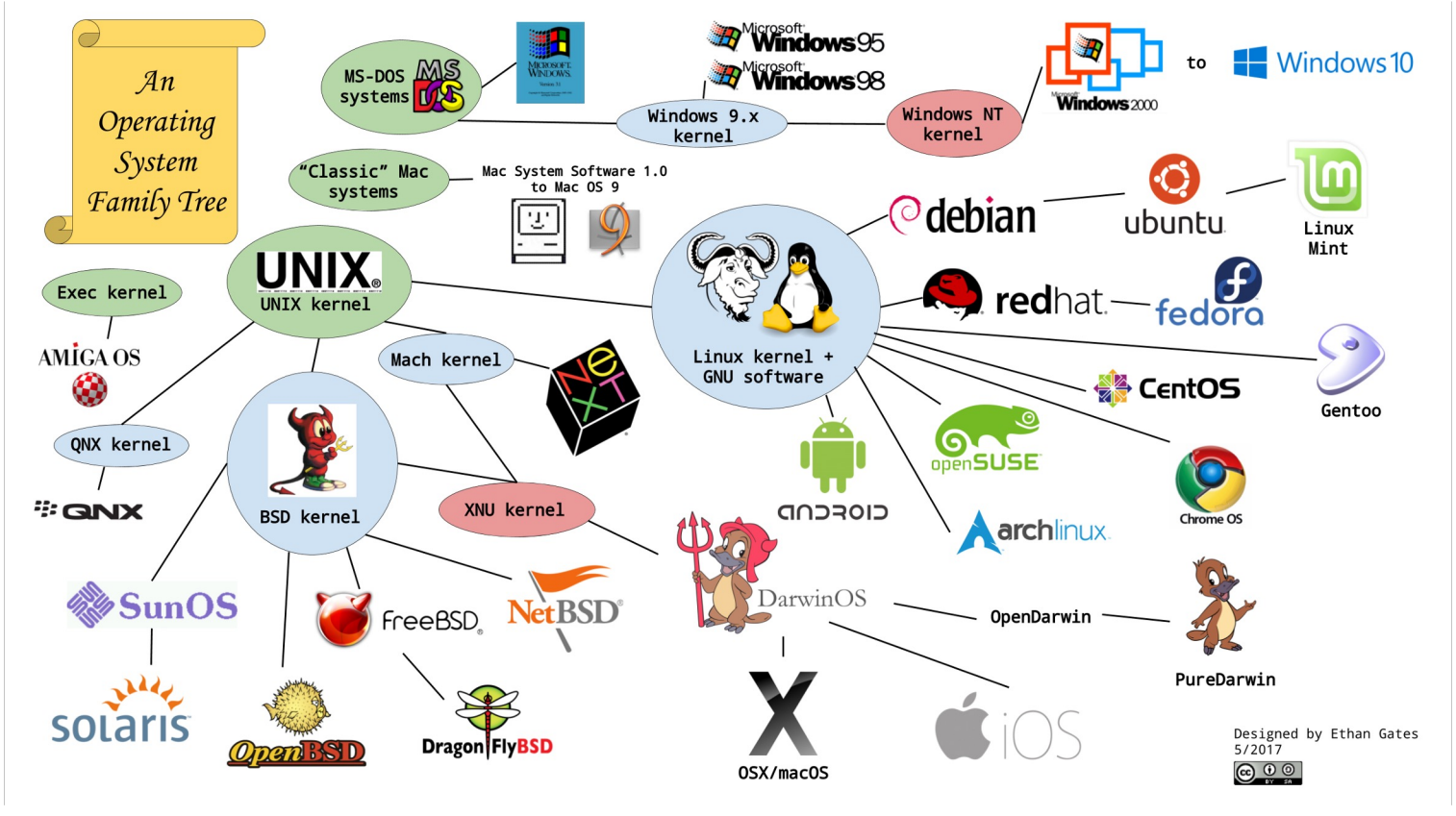
# Hva er egentlig et operativsystem?

# Operativsystemer – Dirigenten i datamaskinen

- Sørge for at flere brukere kan dele en datamaskin
- Sørge for at flere programmer kan kjøre samtidig
- Fungere som et «mellomnivå» mellom maskinvare og programvare



# De fleste dagens operativsystemer er i «slekt»

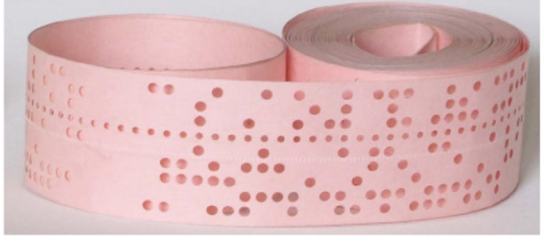


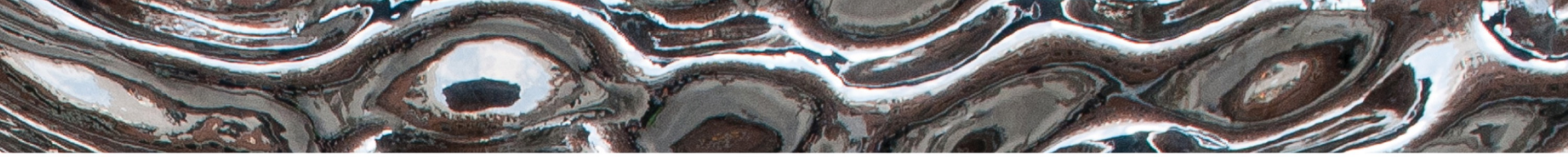
# Kunne vi ha klart oss uten operativsystemer

- *Et eksempel fra «Historieboken»:*

En Nord-12 maskin fra 1972:

1. Lese inn et hullbånd med program for å redigere kode.
2. Lese inn et hullbånd med mitt program.
3. Redigere programmet.
4. Skrive ut et hullbånd med det oppdaterte programmet.
5. Lese inn et hullbånd med kompilatoren.
6. Lese inn hullbåndet med programmet mitt.
7. Kompilatoren skriver et hullbånd med ferdig maskinkode.
8. Lese inn hullbåndet med koden og kjøre programmet.

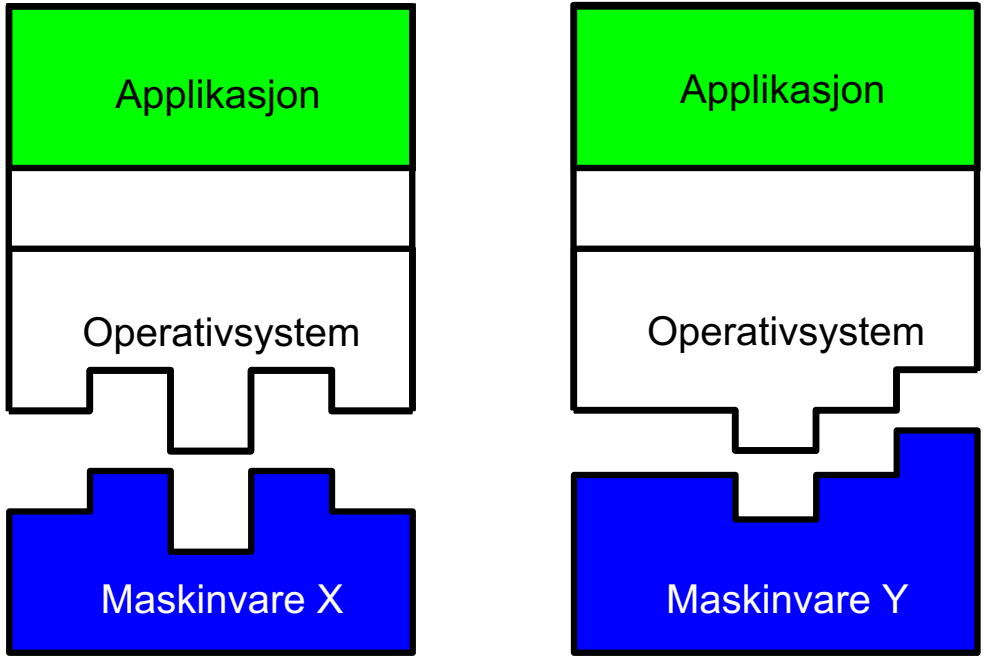




UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

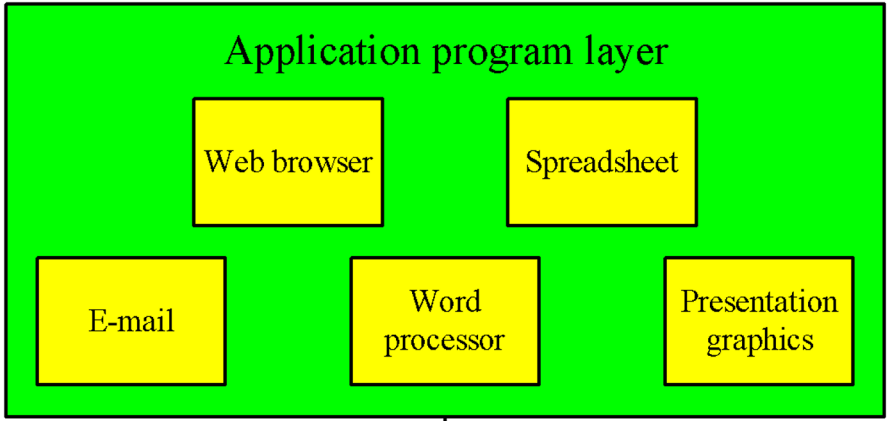
# Operativsystemer:
## Hvordan fungerer de?

# Forskjellig maskinvare



# Mange parallelle oppgaver

- Bedre utnyttelse av maskinen
  - Mange prosesser i parallell
    - Utfører forskjellige oppgaver
    - Bruker forskjellige komponenter
  - Flere samtidige brukere
- Utfordringer
  - "Samtidig" tilgang
  - Beskyttelse / sikkerhet
  - Rettferdighet
  - …





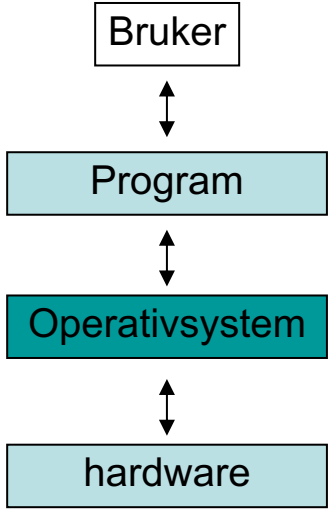

# Hva er definisjonen i litteraturen?

- *"An operating system (OS) is a collection of programs that acts as an intermediary between the hardware and its user(s), providing a high-level interface to low level hardware resources, such as the CPU, memory, and I/O devices. The operating system provides various facilities and services that make the use of the hardware convenient, efficient and safe"*

Lazowska, E. D.: Contemporary Issues in Operating Systems , in: Encyclopedia of Computer Science, Ralston, A., Reilly, E. D. (Editors), IEEE Press, 1993, pp.980

- Det er en enkel maskin (ovenfra og ned)
  - Skjuler kompliserte detaljer
  - Presenterer en «virtuell maskin», enkel å bruke

- Det er en resurshåndterer (nedenfra og opp)
  - Hvert program for sin tid/plass i resursene.



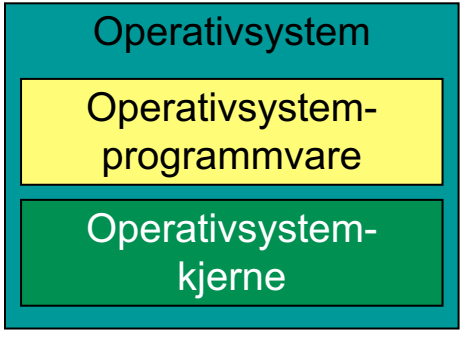
# Hva er definisjonen i litteraturen?

- *"An operating system (OS) is a collection of programs that acts as an intermediary between the hardware and its user(s), providing a high-level interface to low level hardware resources, such as the CPU, memory, and I/O devices. The operating system provides various facilities and services that make the use of the hardware convenient, efficient and safe"*

Lazowska, E. D.: Contemporary Issues in Operating Systems , in: Encyclopedia of Computer Science, Ralston, A., Reilly, E. D. (Editors), IEEE Press, 1993, pp.980

- Det er en enkel maskin (ovenfra og ned)
  - Skjuler kompliserte detaljer
  - Presenterer en «virtuell maskin», enkel å bruke

- Det er en resurshåndterer (nedenfra og opp)
  - Hvert program for sin tid/plass i resursene.



# Hvor finner vi egentlig et operativsystem (OS)?

Datamaskiner

Mobiler

Konsoller

Biler

Kamera,
andre fartøy/båter,
Dekodere,
klokker,
sensorer,
… □ OVER ALT!

# Forskjellige typer operativsystemer (OS)

- Enkel-bruker, enkel-oppgave:
  Historisk og sjelden (kun noen gamle PDA 'er brukte dette)
- Enkel-bruker, multi-oppgaver:
  PCer og arbeidsstasjoner kan være konfigurert slik (typisk gamle mobiltelefoner)
- Multi-bruker, multi-oppgaver:
  brukes på servere, PCer, arbeidsstasjoner, bærbare PCer og de fleste moderne maskiner.

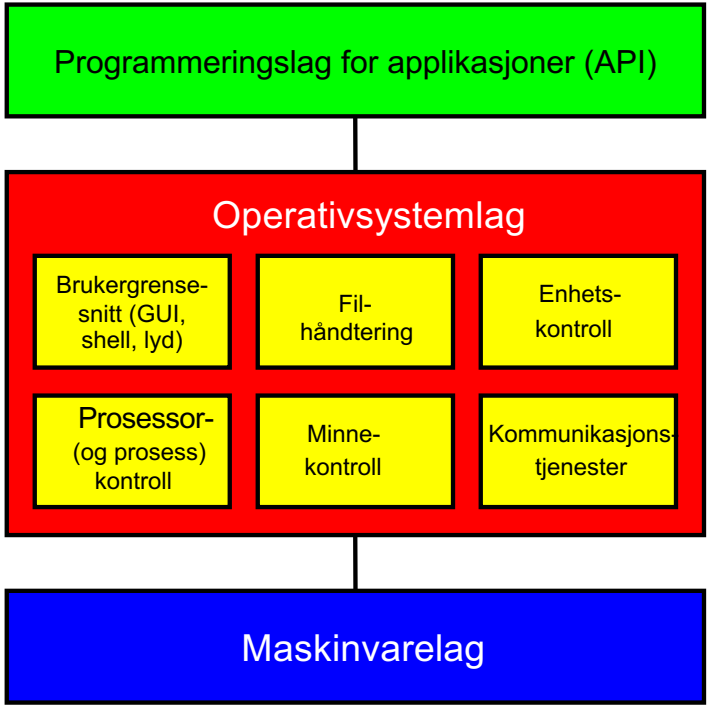
- Distribuert OS:
  Støtte for å kontrollere resurser på flere datamaskiner. Ikke så vanlig.

- Real-time OS:
  Støtte for systemer hvor sanntid er viktig som biler, robotter, fly, instrumenter, atomreaktorer, osv.

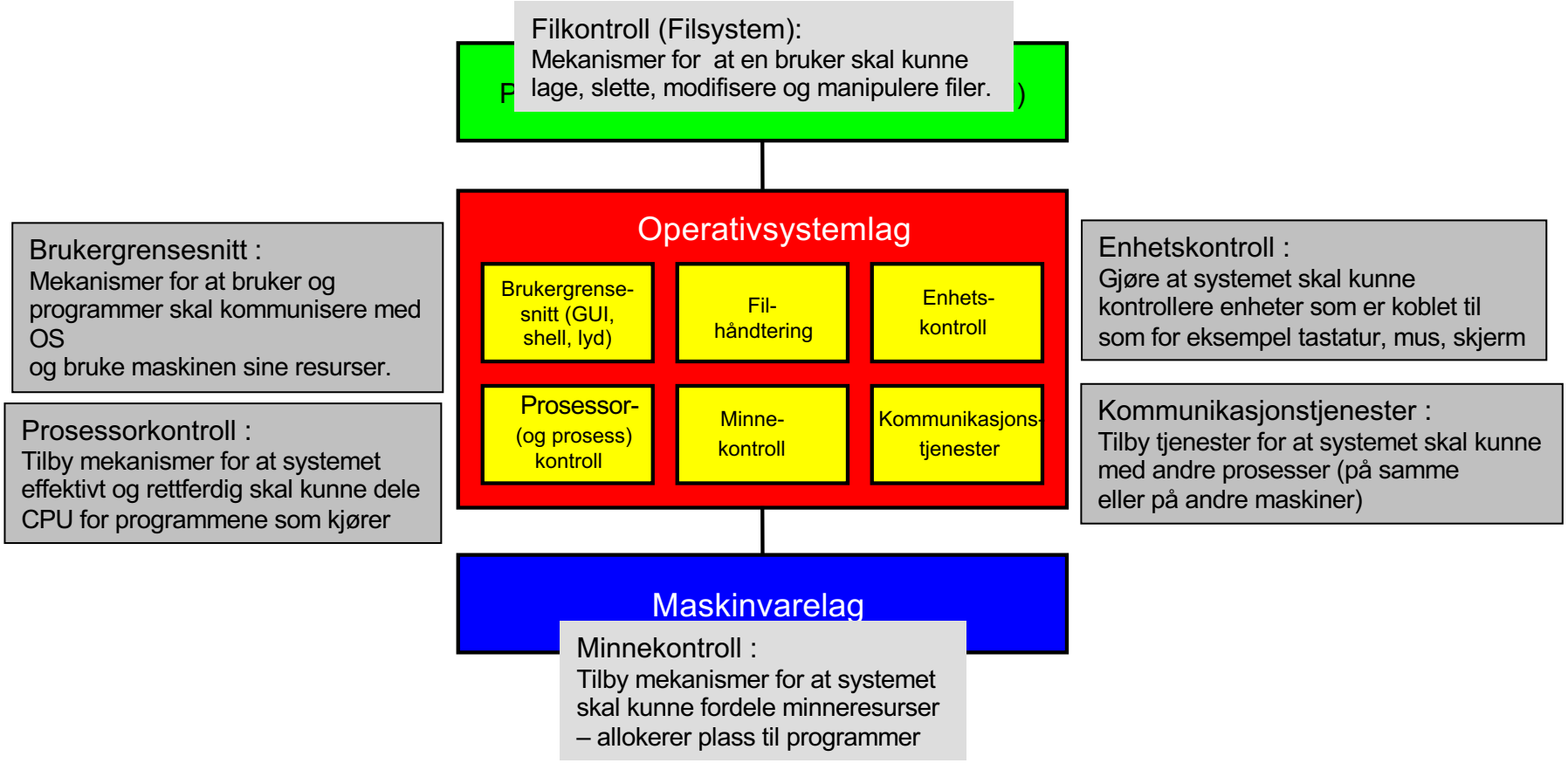
- Embedded OS:
  Bygd inn i enheter for å løse et konkret problem, som for eksempel enkle mobiltelefoner, mikrobølgeovner, vaskemaskiner, dørlås, osv.

# Hovedkomponenter i et operativsystem

- «Synlige" for bruker
  - Brukergrensesnitt
  - Filsystem
  - Enhetskontroll
- "(Relativt)Transparent"
  - Prosessorkontroll
  - Minnekontroll
  - Kommunikasjonstjenester

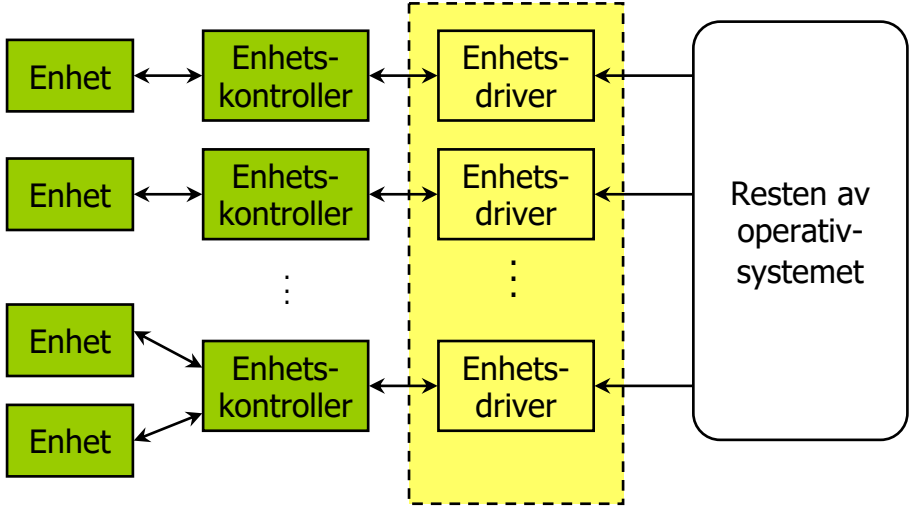


# Hovedkomponenter - Detaljert



# Enhetskontroll

- Operativsystemet må kunne kontrollere enheter tilkoblet maskinen som for eksempel disker, tastatur, nettverkskort, skjermer, høyttalere, mus, USB-minne, kamera, DVD/Blu-ray, mikrofon, skrivere, joysticks, ...
  - Stor forskjell
  - Varierende hastighet
  - Forskjellige måter å hente data



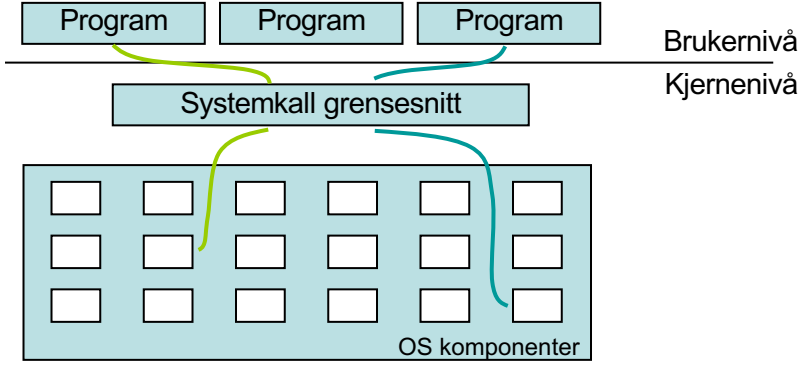
# Brukergrensesnitt

- Koblingen mellom bruker og datamaskinen
- Operativsystemet har logikk som støtter kobling mellom maskinvare og programmer
  - Kommandolinje, for eksempel: en terminal
  - Grafisk brukergrensesnitt (GUI): Grensesnitt med vinduer, ikoner, menyer og peker
  - Selve grensesnittet ligger ikke i operativsystem**kjernen**
- Eksempel: ***X*** (i Linux se `man X`)
  - Vindushåndteringssystem som kjører på de fleste ANSI C og POSIX (Portable OS Interface for UNIX) kompatible systemer
  - Bruker kommunikasjon mellom prosesser for å få input fra, og sende output til flere programmer
- Andre: Desktop Window Manager (Windows), Wayland (Linux)

# Systemkall

- Grensesnittet mellom operativsystemet og programmer (bruker) er definert med hva vi kaller et systemkall

- Når operativsystemet gjør et systemkall er det ganske likt at vi kaller en funksjon i Python, men systemkallet går inn i kjernen på operativsystemet:



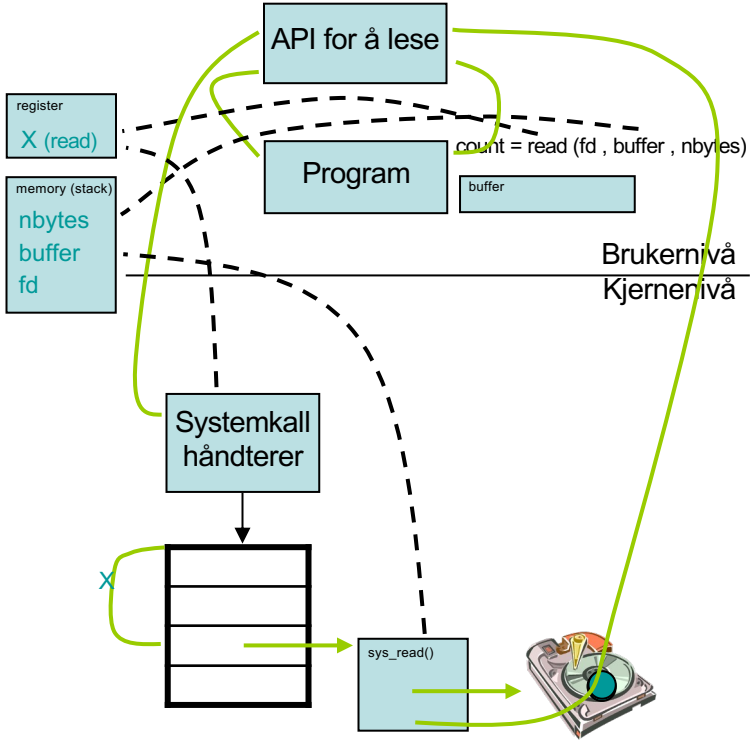

Linux:
x86 v2.4.19 entry.S □ 242
x86 v3.0-rc4 syscall_table_32.S □ 347
x86 v3.16.2 syscall_32.tbl □ 353

FreeBSD:
v9 syscalls.c □ 531

# Eksempel på systemkall: `read`

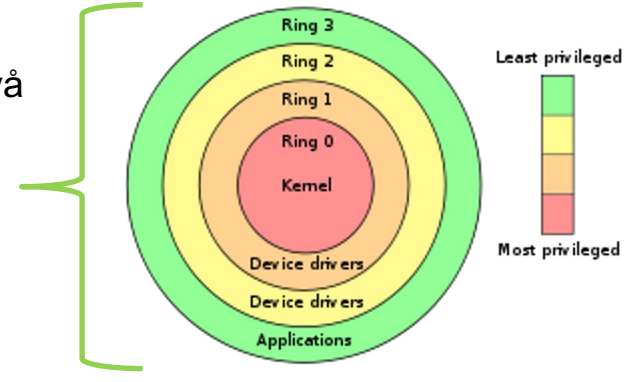
- Eksempel i programmeringsspråket C:
  **`count = read(fd,buffer,nbyte)`**

1. Lagrer parametere i minne
2. Kaller API som programmet bruker
3. Setter systemkall nummer i minne
4. Kaller kjernen (TRAP)
   - Kjernen undersøker systemkall nummer
   - Kjernen finner hvor kallet skal sendes
   - Kjernen utfører operasjonen
5. Tilbake til API for å rydde opp
   - Fjerner parametere fra minne
6. Fortsetter programmet



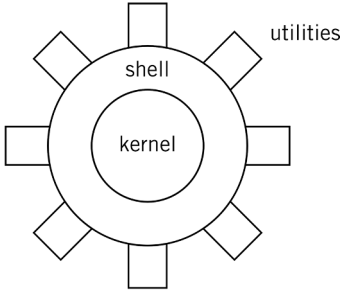
# Brukernivå vs. Kjernenivå (Beskyttelse)

- Mange OS skiller mellom brukernivå og kjernenivå, grunnen til dette er sikkerhet og beskyttelse
- Operativsystemet kjører i kjernemodus
  - *Real mode*
  - Tilgang til hele minne
  - Alle instruksjoner kan bli kjørt
  - Går forbi all sikkerhet
  - Bytte til *Protected mode* eller *Long mode* snarest mulig
- Vanlige programmer og flere API'er kjører i brukernivå
  - *Protected mode*
  - Bruke privilegerte nivåer (x86: «ring», ARM: «mode»)
  - Ytre ringer
    - kan ikke kontakte maskinvare eller enheter direkte, kun gjennom et API
    - tilgang kun til et begrenset minneområde
    - begrenset tilgang til prosessoren

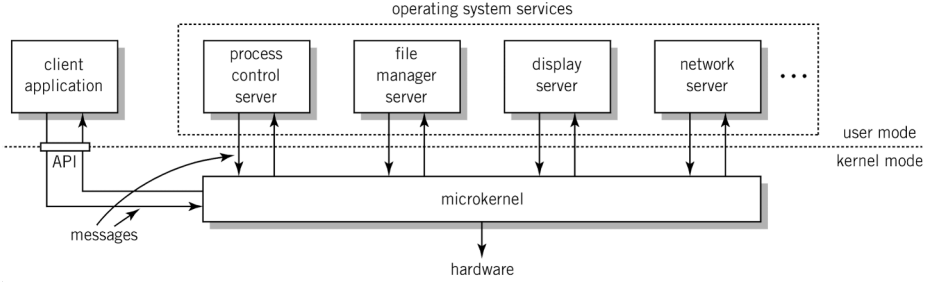


# Organisering av operativsystemer

- Det er ikke noe bestemt standard for hvordan et operativsystem er bygd opp (i motsetning til nettverksprotokoller, etc.), men det er to hovedmodeller:




- Monolittisk kjerne ("the big mess"):
  - All beskyttet kode kjører i samme beskyttelsesområde
  - Effektivt: alle komponenter kan kommunisere direkte (funksjonskall, skrive i minne)
  - Feil kan få følger for andre komponenter enn der de oppstår
  - Store og komplekse
  - BSD UNIX, Linux, Windows 10 (++), …
- Mikro-kjerner
  - Små moduler med minimal funksjonalitet (avbrudd, minne, prossessor, osv.)
  - Så mange som mulig av tjenestene er skriver i brukermodus
  - Mye beskjeder frem og tilbake (ineffektivt)
  - Vanskelig å utføre ting i riktig rekkefølge
  - Vanskelig feilsøking pga grensene
  - Liten, modulær, utvidbar, portabel, …
  - MACH, Windows NT, …



# Oppsummering

- Operativsystemer skjuler kompleksiteten i maskinvare
- Tillater flere brukere å bruke maskinen samtidig
- API for å tillate programmer å bruke resurser i maskinen
- Grunnleggende sikkerhetsmekanismer er bygd inn

- Flere detaljer om kommunikasjonstjenestene i slutten av oktober! ☺

# Ekstramateriale:

- *Bøker og artikler:*
  - Andrew S. Tanenbaum, Herbert Bos: *Modern Operating Systems (4th edition),* 2015. Pearson

- *Kurs om operativsystemer på IFI:*
  - IN2140 – Introduksjon til operativsystemer og datakommunikasjon
  - IN3000 – Operativsystemer