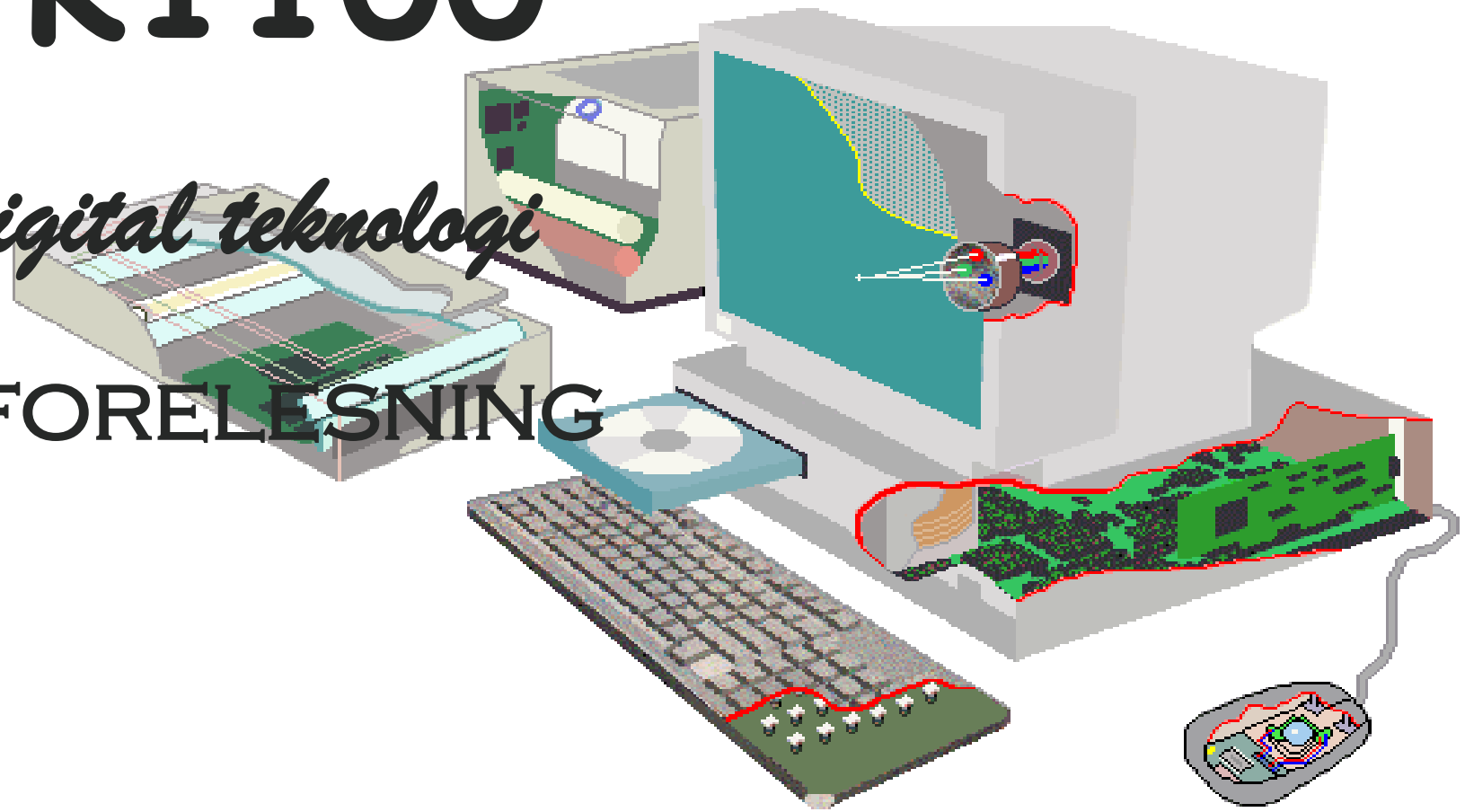
TK1100

Digital teknologi

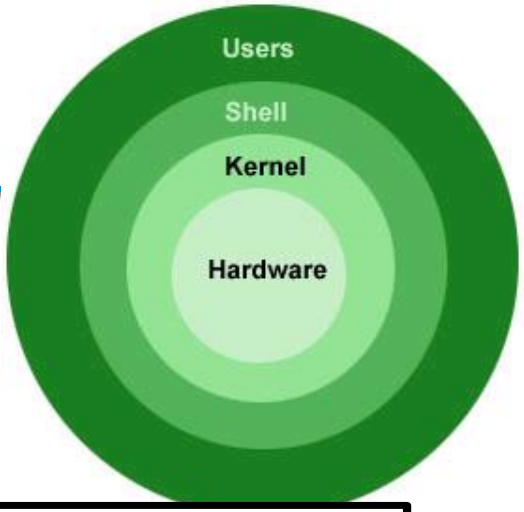
5. FORELESNING



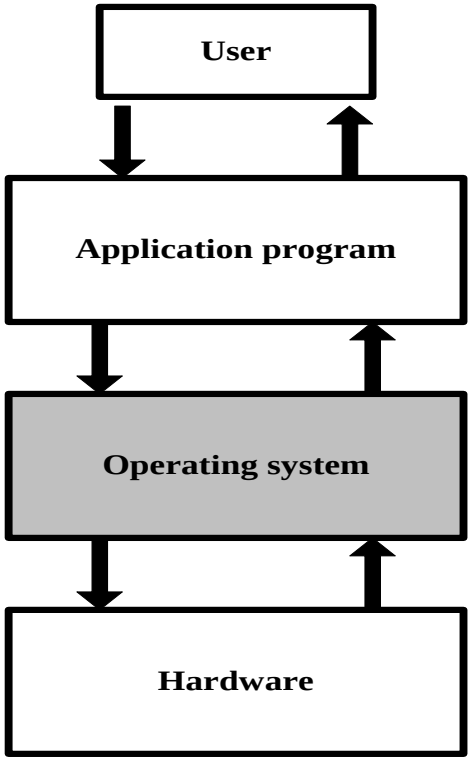
I dag

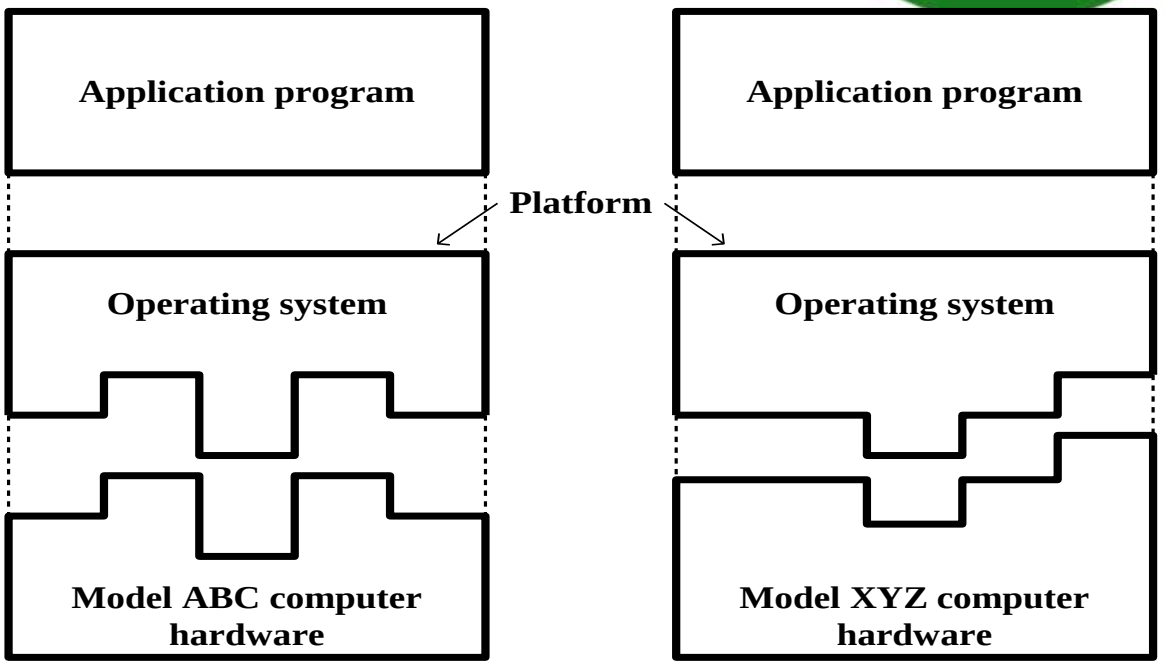
- Fokusere på å få en *oversikt*
- Lære *begrepene* som benyttes i litteratur og dokumentasjon
- Øving
  - Teori og begreper, samt litt historie
  - Lære seg kommando-linje interface på eget OS, hvordan manøvrere seg og hvordan gjøre enkle oppgaver i kommando-linje

Hva er et operativsystem?



Operativsystemet er programvare som ligger mellom brukeren/programmereren og maskinvaren





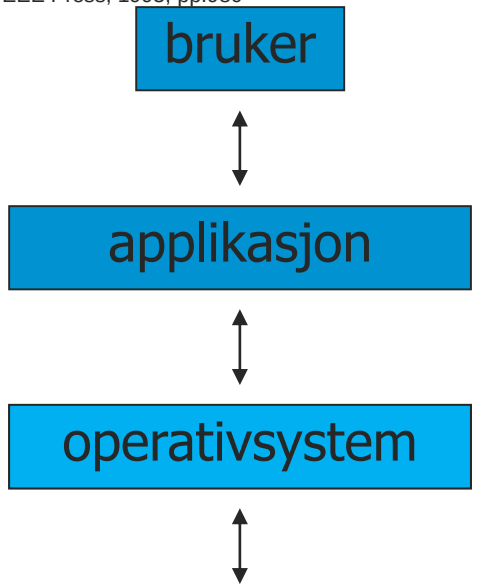
Samme applikasjon på ulike typer maskinvare (OS = “plattform”)

Hva er et operativsystem (OS)?

- *"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 utvidet/abstrakt maskin (top-down view)
  - Skjuler de "grisete" detaljene i HW
  - Tilbyr brukeren og programmereren en virtuell maskin som det er enklere å programmere/bruke
- Det er en ressursadministrator (bottom-up view)
  - Hvert program får tid på ressursen
  - Hvert program får plass på ressursene (CPU, Minne,…).



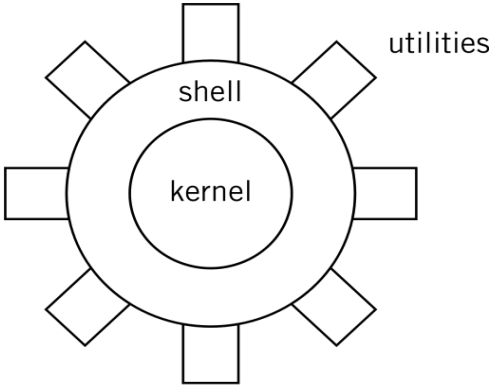
Abstraksjon

- Å abstrahere vil si å ta bort (uvesentlige) detaljer.
- For eksempel: En fil vil (oftest) bestå av metadata (data om data)
  - Et navn
  - når opprettet, sist endret, o.l.
  - hvor stor (KiB)
  - hvilke sektorer på disken data er fysisk plassert på
  - ... og selve innholdet (data/instruksjoner)
- Filene organiseres i kataloger (directories/~~mapper~~)
  - en annen abstraksjon, som også er en type fil som inneholder metadata og adresser til andre filer
- **Fysisk**/konkret er den bare magnetiserte områder i adresserte sektorer på en disk/CD/minnepinner
- For brukeren fremstår denne abstraksjonen som symboler h@n kan klikke på.
- For programmeren tilbyr OS systemkall som gjøre at h@n kan opprette, åpne, lukke, posisjonere seg innenfor, lese fra/skrive til

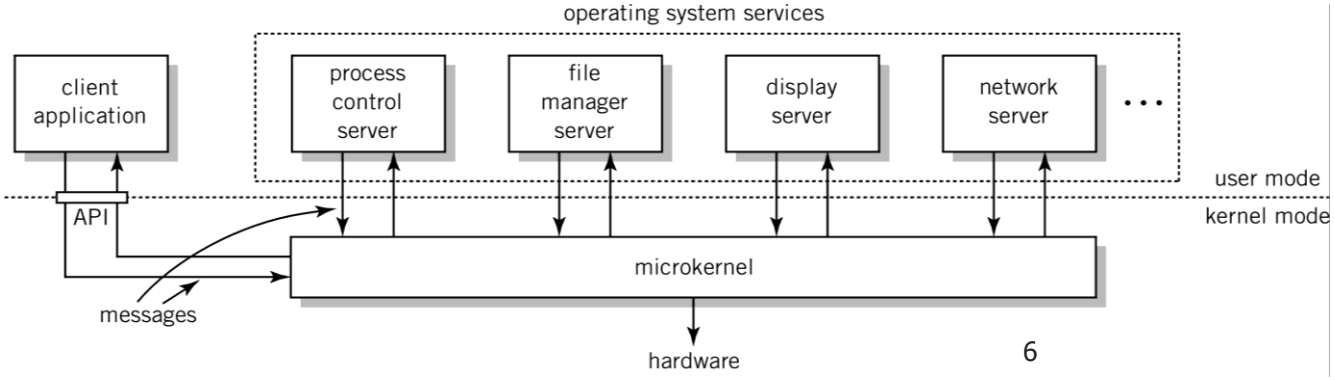
OS Organisering

- Flere mulig tilnærminger, ingen standard.

- Monolithic kernel ("the big mess"):
  - Skrevet som en samling av funksjoner som er linket sammen til ett objekt.
  - Vanligvis effektivt (ingen grenser som krysses i kjernen)
  - Store, komplexe, kræsjer rimelig lett
  - UNIX, Linux, Windows NT, OSX (i praksis, men MACH i starten)



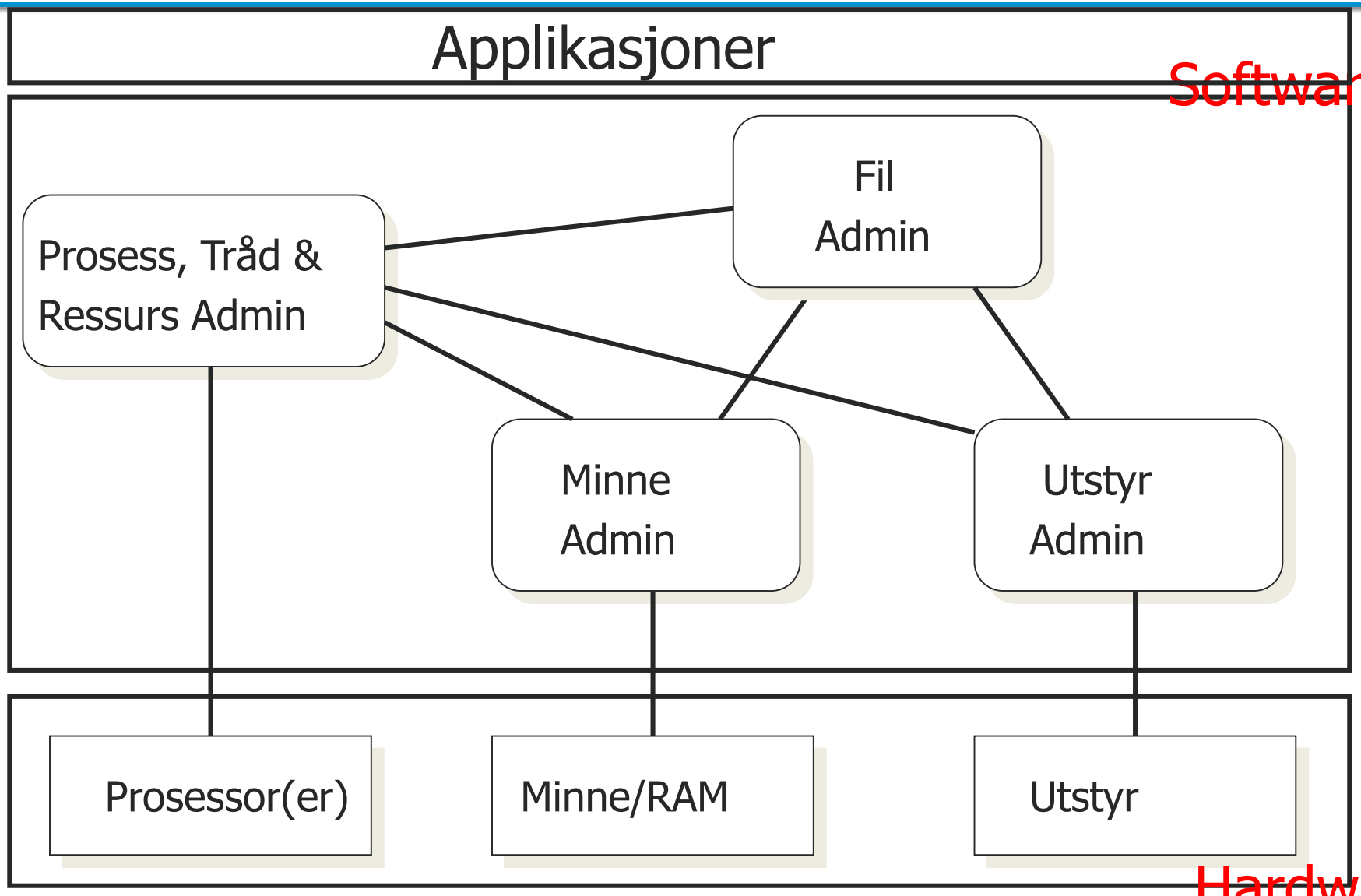
- Micro kernel
  - Kjerne med minimal funksjonalitet (administrere interrupt, minne, prosessor)
  - Andre tjenester implementeres som server prosesser i brukermodus i henhold til en klient-tjener-modell.
  - Mye meldingsutveklsing (ineffektivt)
  - lite, modulært, utvidbart, portablet, …
  - MACH, L4, Chorus, …



Funksjoner i operativsystemer

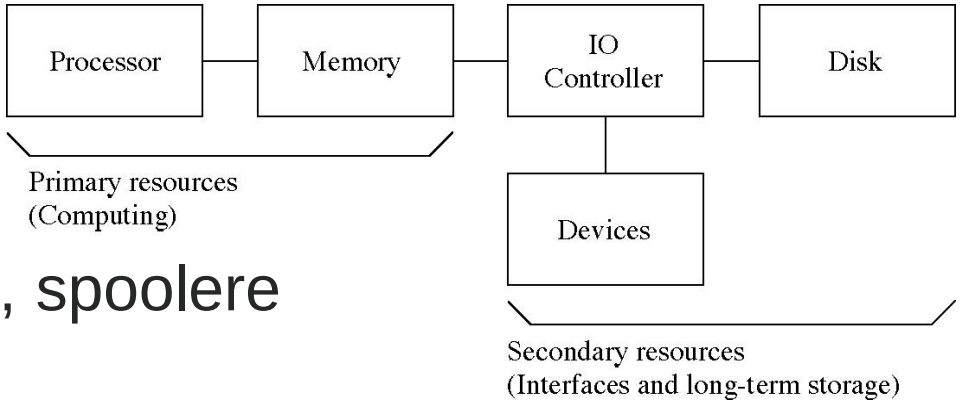
- Brukergrensesnitt (skall!)
- Applikasjons-kjøring
  - Tilbyr API (Application Programming Interface)
  - Tilbyr SPI (System Programming Interface)
- Håndtering av ressurser
  - Prosesser, hukommelse, eksterne lager, I/O-enheter
- Håndtering av maskinvare
  - Drivere!
- Håndtering av nettverk
- Sikkerhet
  - Oftest tett knyttet opp til **filsystemet**
- **Ikke alle** anvendelser av datamaskiner trenger et OS

Analyse av OS «kjerne»



Operativsystemet - hovedfunksjoner

- Oppstart av maskinen
- Intern behandling
  - GUI, **«auditing»**, sikkerhet, feilbehandling
- Prosess behandling
  - Oppstart, fjerning, scheduling, responstid
- Minne behandling
  - Paging, segmentering, virtuelt minne, delt minne
- Disk behandling
  - Filsystem
- Device/Utstyr behandling
  - Drivere, interrupt behandling, spoolere
- Nettverk behandling
- Service-programmer (utilities)
  - Filbehandler, backup, defragmentering .....



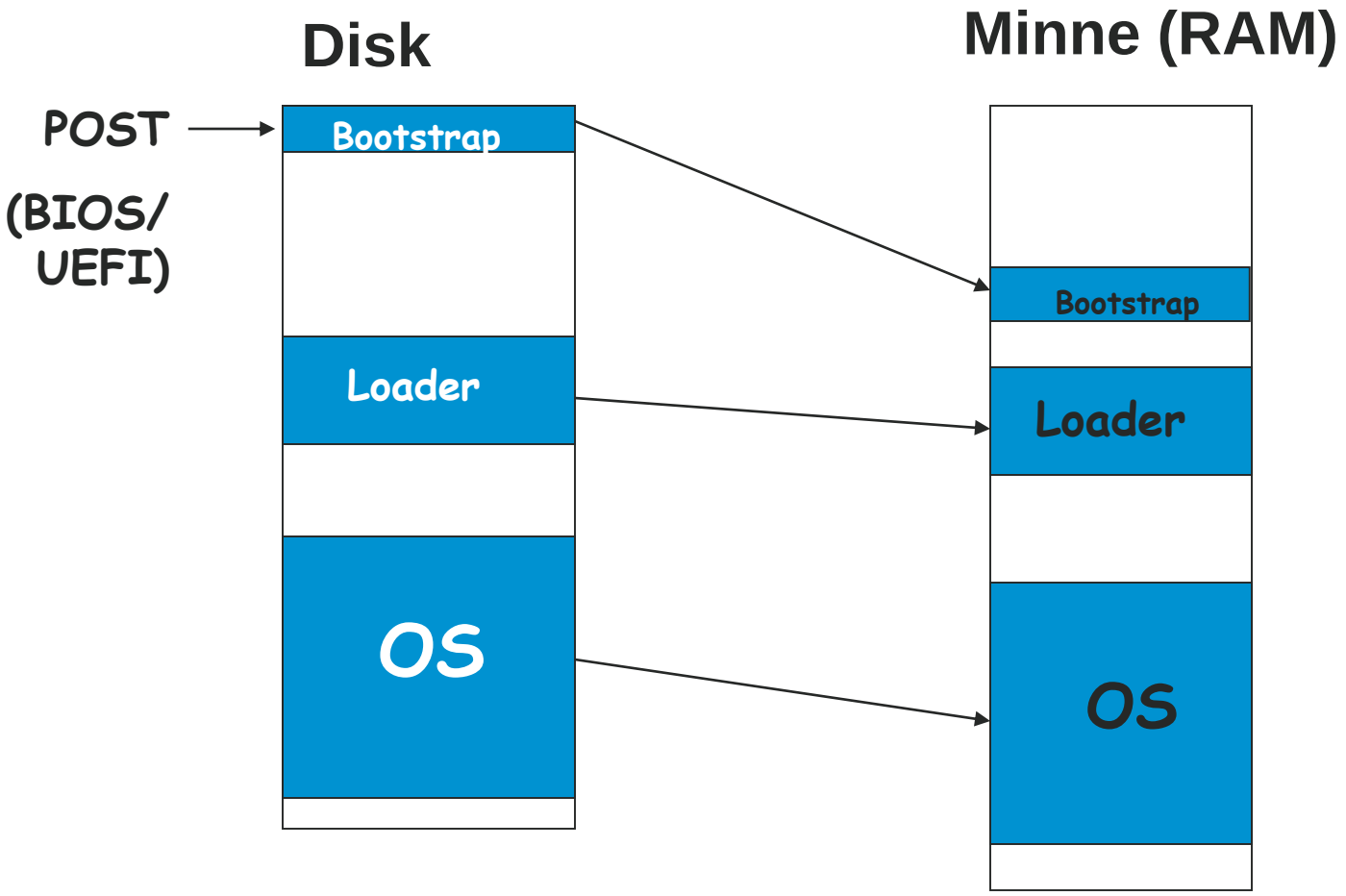
TEORI

Kjernen starter opp og konfigurer hardware

Kjernen administrerer prosesser, tråder og russerser

PROGRAMMERER-PERSPEKTIV: KJERNEN («KERNEL»)

Oppstart (bootstrapping)



1. Hent bootstrap
2. Hent Boot-loader
3. Hent OS
4. Initialiser maskinvare
5. Lag brukergrensesnitt
6. Start faste programmer

Begreper

- Prosess
  - Et program under kjøring med tilhørende ressurser
- Tråd
  - ”Thread of control”– selve kjøringen av instruksjoner (en og en…)
- Ressurs
  - Alt et program trenger for å kjøre ferdig.
    - CPU, RAM, I/O, …

Bruker- vs kjernemodus

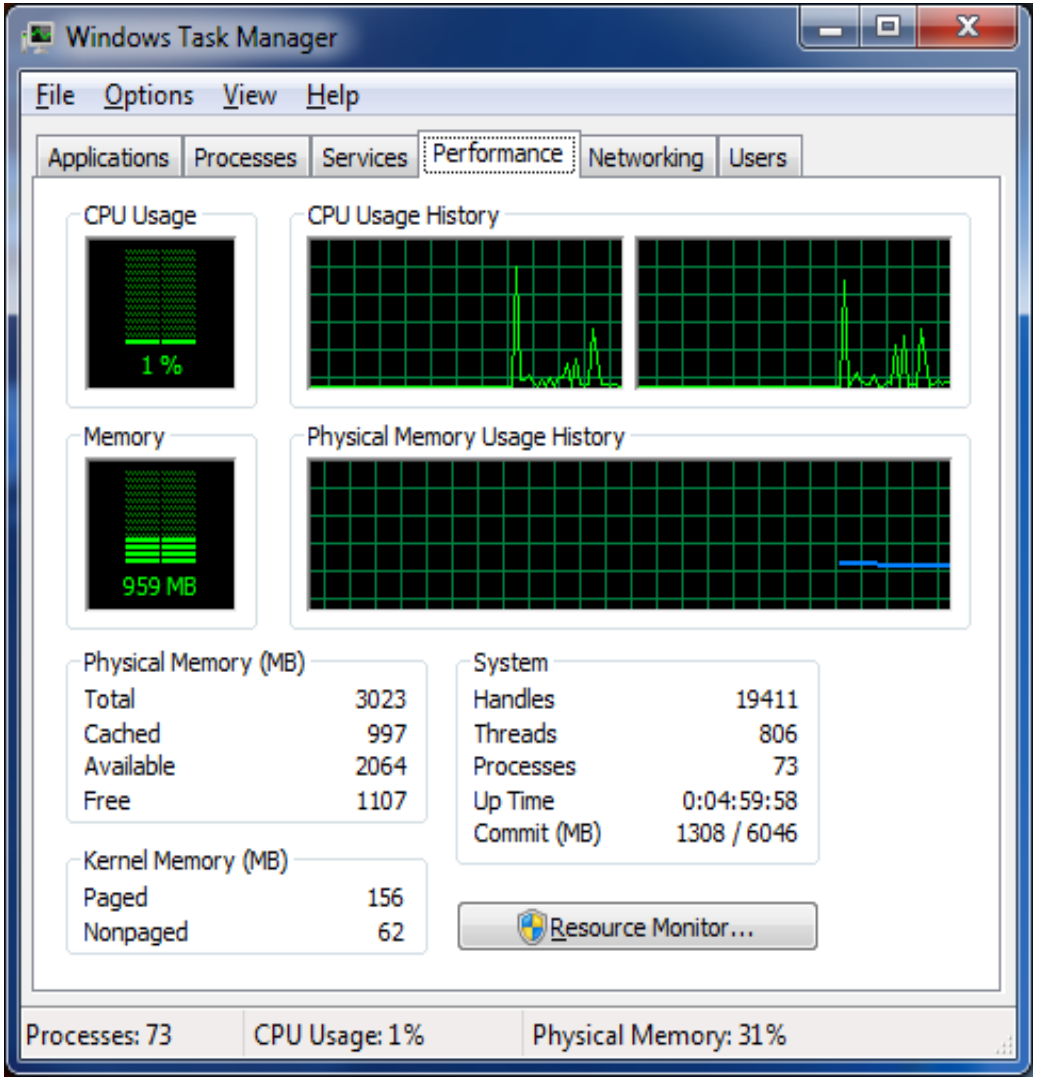
- For å oppnå sikkerhet og beskyttelse gir de fleste CPUer muligheten til å kjøre instruksjoner enten i **applikasjons**- eller **kjerne-modus**

- **Vanlige applikasjoner** og mange OS-tjenester og kjører i "**user mode**" (Intel/AMD "ring 3")
  - Kan ikke aksessere HW, utstyrsdrivere direkte , må bruke en API
  - Kun adgang til minnet som OSet har tildelt
  - Begrenset instruksjonssett

- OS kjører i kjernemodus ("ring 0")
  - Tilgang til hele minnet
  - Alle instruksjoner kan kjøres
  - Ingen sikring fra HW

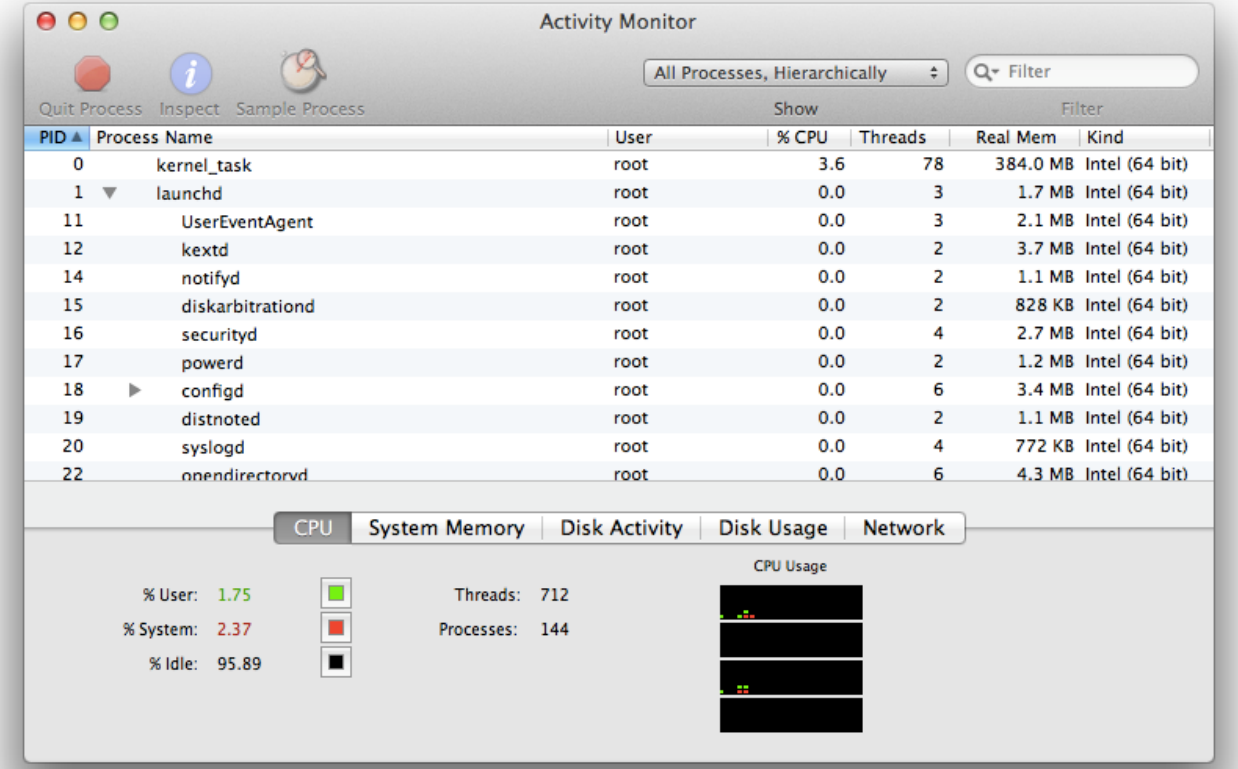
Prosesser og tråder: Taskmanager

- Windows – Taskmanger
  - Ctrl-Shft-Esc
  - Lar deg inspisere prosesser og deres bruk av ressurser



OSX – Activity Monitor

- Viser (sorterte) oversikter over prosesser, tråder, ressurser.
- Kan også bruke bash-verktøyene…



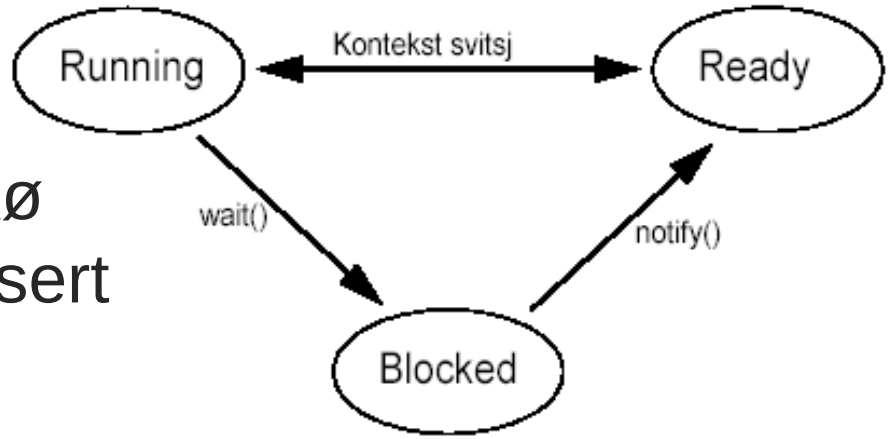
Linux: ps, top, mmfl

- Linux /OSX – ps, top og /proc
  - `ps` – kommando
  - `top` – applikasjon
  - `/proc` – del av ”filsystemet”

```
top - 17:14:56 up 138 days,  2:02,  3 users,  load average: 0.00, 0.00, 0.00
Tasks:  72 total,   2 running,  70 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni, 96.7%id,  3.3%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   2059632k total,  1587720k used,   471912k free,   342932k buffers
Swap:  2096472k total,       72k used,  2096400k free,   855740k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0 10348  632  536 S  0.0  0.0   0:01.58 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.04 ksoftirqd/0
    4 root      10  -5     0    0    0 S  0.0  0.0   3:27.07 events/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
   22 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
   26 root      10  -5     0    0    0 S  0.0  0.0   0:00.26 kblockd/0
   27 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
  186 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
  189 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
  191 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kseriod
  259 root      10  -5     0    0    0 S  0.0  0.0   0:14.39 kswapd0
  260 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 aio/0
  466 root      11  -5     0    0    0 S  0.0  0.0   0:00.00 kpsmoused
  496 root      10  -5     0    0    0 S  0.0  0.0   0:30.63 mpt_poll_0
  497 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 scsi_eh_0
```

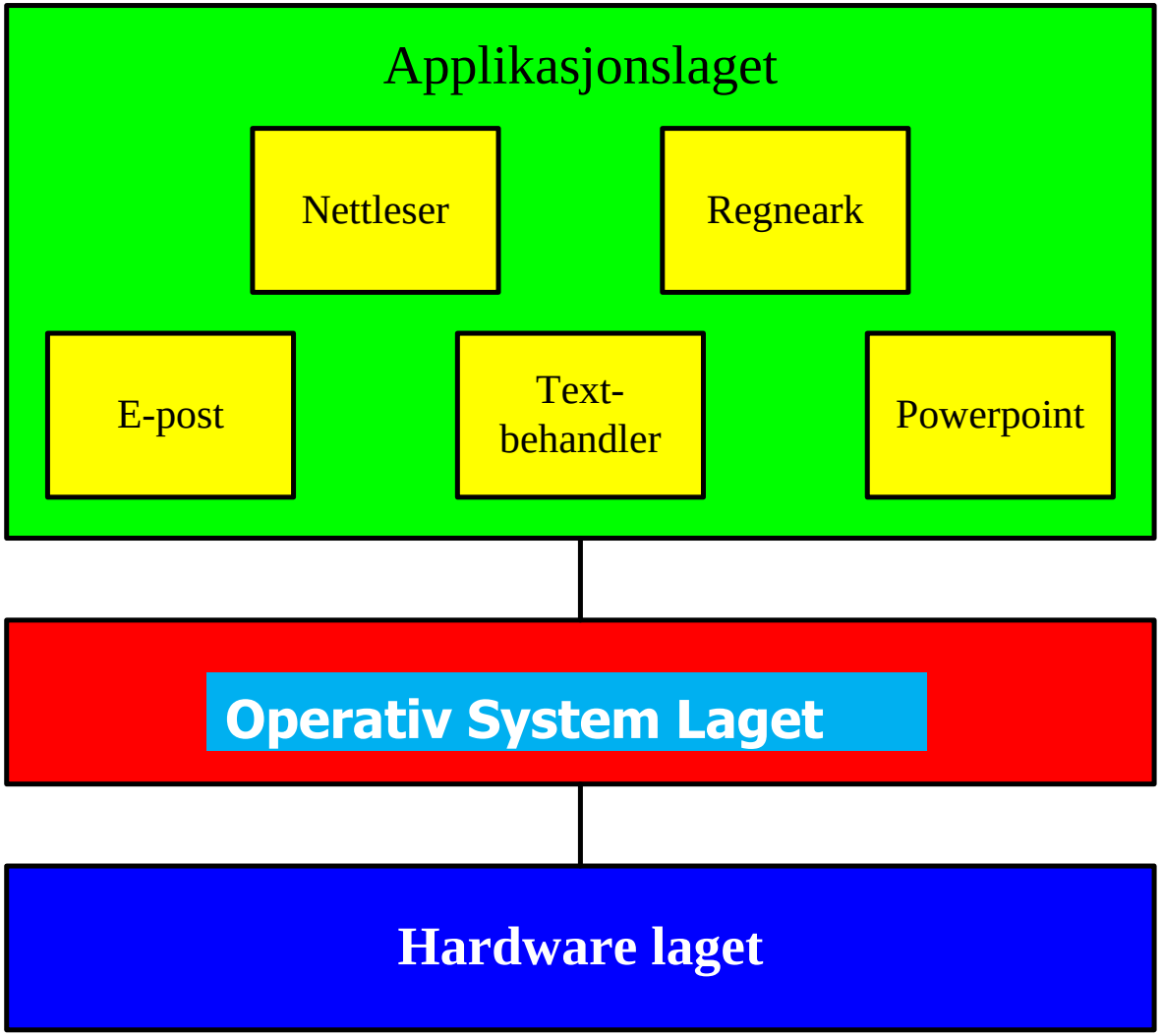
Operativsystemet - multitasking

- Datamaskinen kan kjøre flere prosesser «samtidig»
- CPUen kan bare behandle en instruksjon fra en prosess til enhver tid (på hver prosessorkjerne)
- Operativsystemet må derfor holde styr på hvilken prosess som skal få benytte CPUen
- CPU-bytte mellom prosesser heter context swiching
  - Running: Kjører nå
  - Ready: Aktivisert og venter i kø
  - Blocked: Venter på å bli aktivisert



Mange Samtidige Oppgaver

- Bedre utnyttelse
  - mange samtidige prosesser
    - Utfører ulike oppgaver
    - Bruker ulike deler av maskinen
  - mange samtidige brukere
- Utfordringer
  - "samtidig" tilgang
  - beskyttelse/sikkerhet
  - rettferdighet
  - …

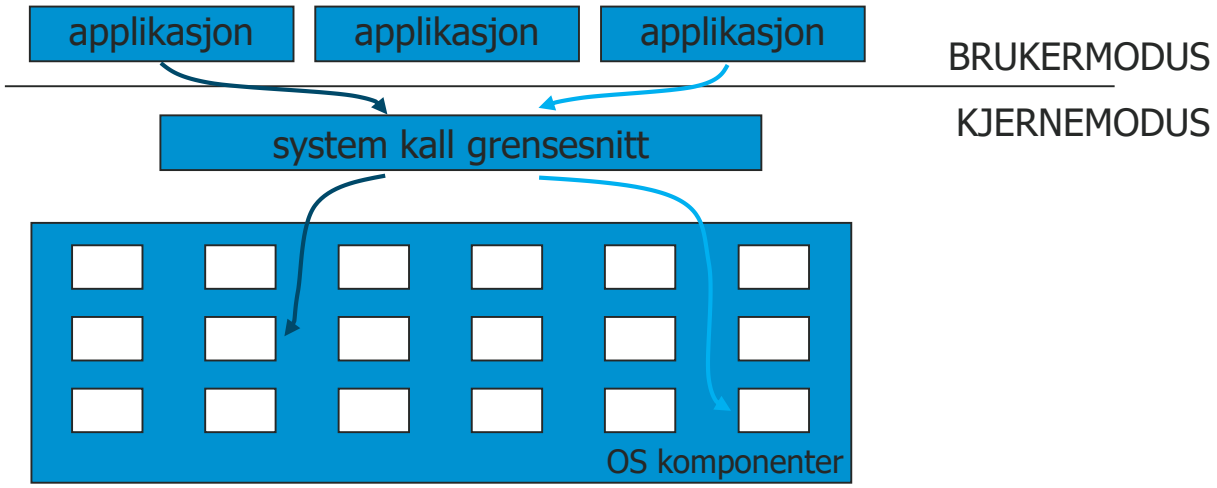


Systemkall

Linux system kall

- Grensesnittet mellom OS og brukerene defineres med en mengde **systemkall**

- Å utføre et systemkall er tilsvarende et vanlig metodekall, men systemkallet kjører kjerne-kode:



Prosess opprettes (Unix) – `fork()`

•Prosess 1

•Prosess 2

rett etter `fork()`

etter terminering
(eller når som helst)

- Protocol control block (process descriptor)
- PID
- address space (text, data, stack)
- state
- allocated resources

rett etter `fork()`

Operativsystemet - minne

- Programmer som kjøres ligger i minnet
  - Operativsystemet er også et program!
- Vanlig RAM-minne er forbedret på flere måter
- Cache er en meget raskt minne (SRAM) som brukes som mellomlager
  - Brukes både mellom CPU og RAM og for enkelte devicer
- Hvis RAM-minnet er for liten, kan noe av innholdet midlertidig legges på disk («swapping»)

Virtuelt minne

- = paging **og** swapping
- Programmet kjøres instruksjon for instruksjon
  - Ikke alle instruksjoner trenger å være til stede i minnet, bare de som skal kjøres
- Abstraher minnebruken!
  - Programmet selv tror det har hele minnet (alle adressene) tilgjengelig.
  - Programmet kompileres med interne (logiske) minneadresser…
  - Programmet deles opp i sider (pages) som bare lastes inn i minnet dersom adressen siden inneholder refereres til.
    - Referanse til en page som ikke er lastet i minnet utløser en PAGE FAULT
  - CPU har en MMU (Memory Managment Unit) som oversetter mellom program-interne (logiske) adresser og faktiske fysiske adresser i RAM (page table)
- Dersom minnet blir for fullt kan minne-sider legges ut på disk (swapping).

Kjøreplan (schedule)

- OS bruker en *kjøreplan* for å bestemme hvilken tråd som skal få lov å kjøre på prosessoren
- Kontekst-svitsjing tar tid
- Det finnes to overordnede kjøreplan-prinsipper
  - Ikke-preemptiv (frivillig)
    - Tråden som er *running* setter seg selv til *ready* (yield-instruksjon)
    - Høflighets-metoden
  - Preemptiv
    - Et klokkestyrt avbrudd sørger med jevne mellomrom for at den tråden som er *running* settes til *ready*
    - Power-metoden
- NT, OSX og Linux har preemptive kjøreplaner basert på prioritets-køer og Round Robin (”Ta den ring”) algoritmen

Kompilering

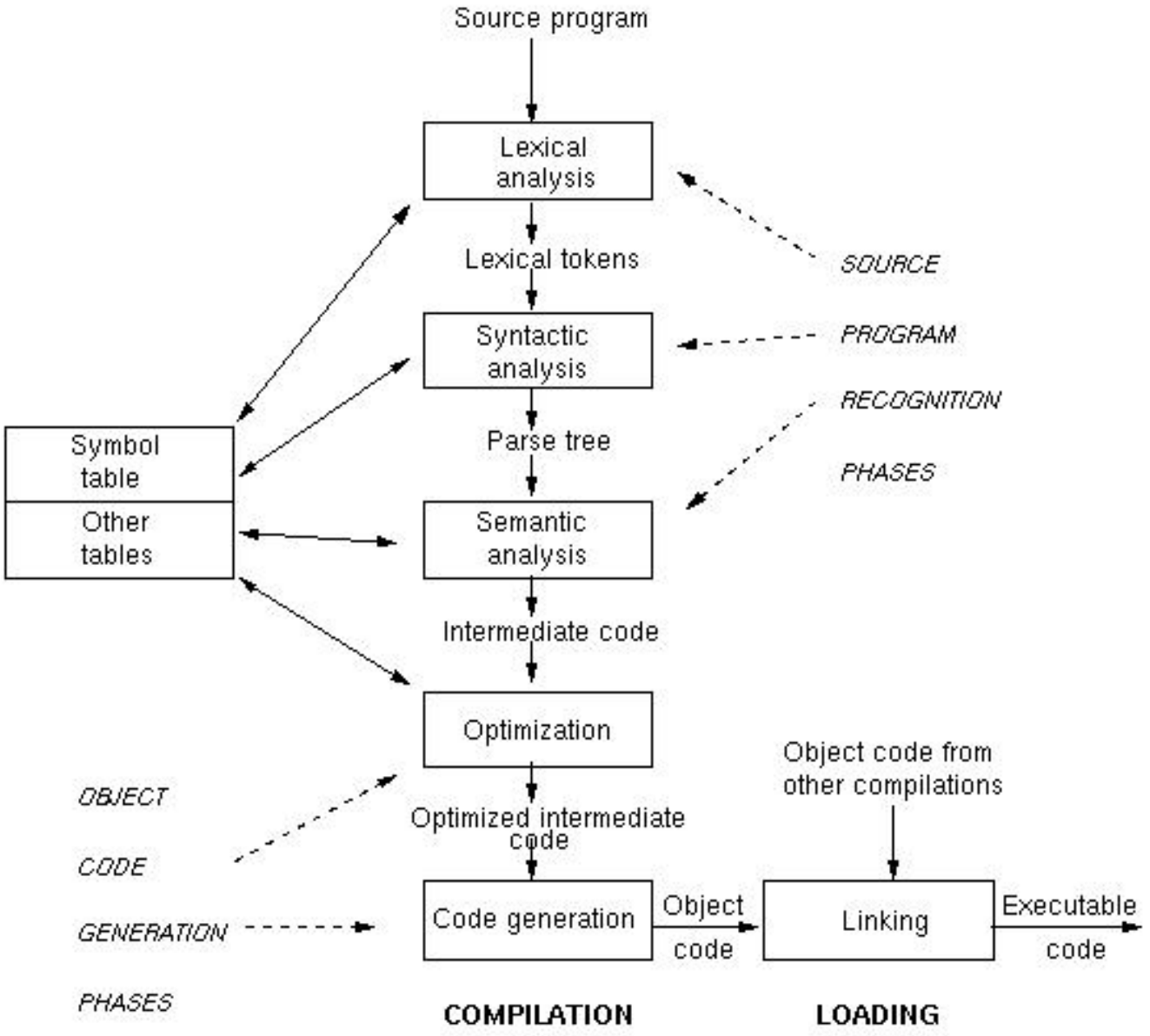
i C

```
#include <stdio.h>

int main() {
    printf("Hello World");
    return 0;
}
```

Assembly

```
MOV AH, 09
MOV DX, 109
INT 21
INT 20
DB "Hello World!","$"
```

```
B4 09 BA 09 01 CD 21 CD
 20 48 65 6C 6C 6F 20 57
6F 72 6C 64 21 24

 ......!. Hello W
 orld!$......4.T.
```



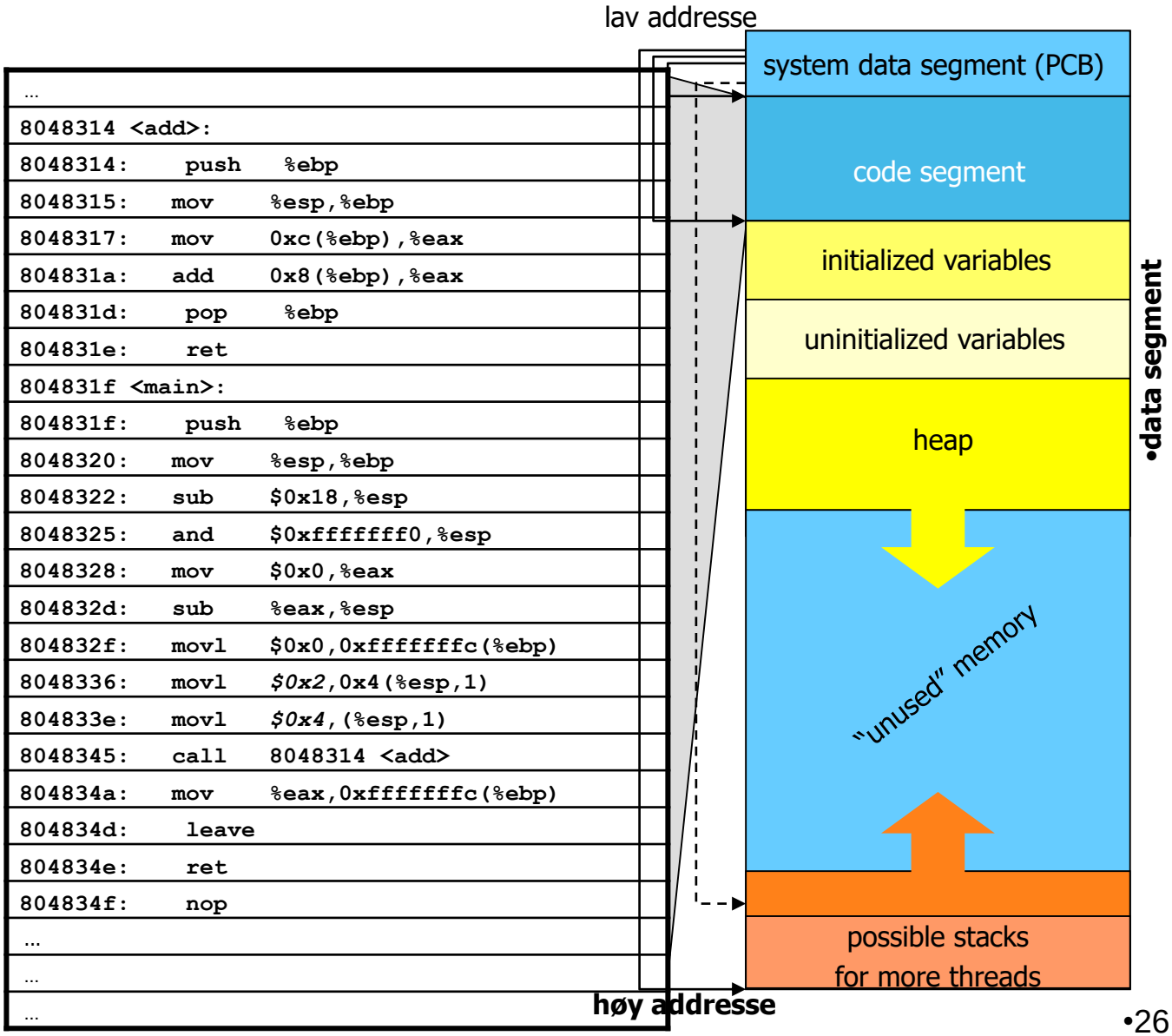
Ulike typer eksekverbare filer

- Ulike OS benytter ulike formater for kjørbare programfiler
  - Windows: Portable Executable (PE)
  - Linux: ELF
  - OSX: Mach-O
- De benytter også forskjellige systemkall og standard biblioteker

Prosessen (tråden) sitt Minne

På Intel arkitekturen partisjonerer en oppgave (task) sitt tildelte minne

- et text (code) segment
  - Lest fra program fil for exempel av `exec`
  - vanligvis read-only
  - Kan deles av flere tråder
- et data segment
  - initialserte globale variabler (0 / NULL)
  - uinitaliserte globale variabler
  - heap
    - dynamisk minne f.eks., allokert med `malloc`
    - vokser oppover
- et stack segment
  - Variabler i en funksjon
  - Lagrede registertilstander (kallende funksjons EIP)
  - Vokser nedover
- system data segment(PCB)
  - segment pekere
  - pid
  - program and stack pekere
    - …
- Flere stacker for trådene

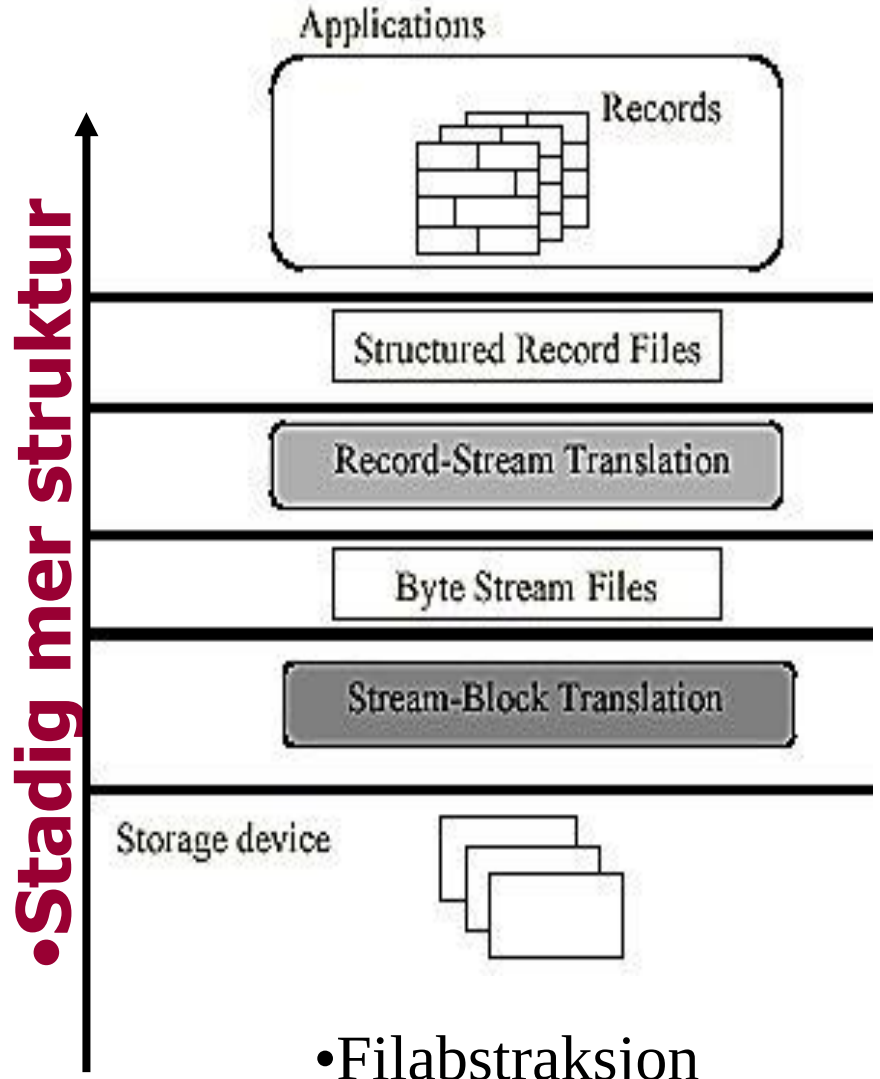


Operativsystemet - filer

- Data som ligger i en hukommelse forsvinner når strømmen slås av
- Data kan også lagres permanent
  - Disk, diskett, CD, tape, ...
- Et slikt lager består av mapper og filer
  - Hierarkisk struktur
  - Navn, tilgangs-rettigheter, skapelses-dato, ...

Filadministrasjon

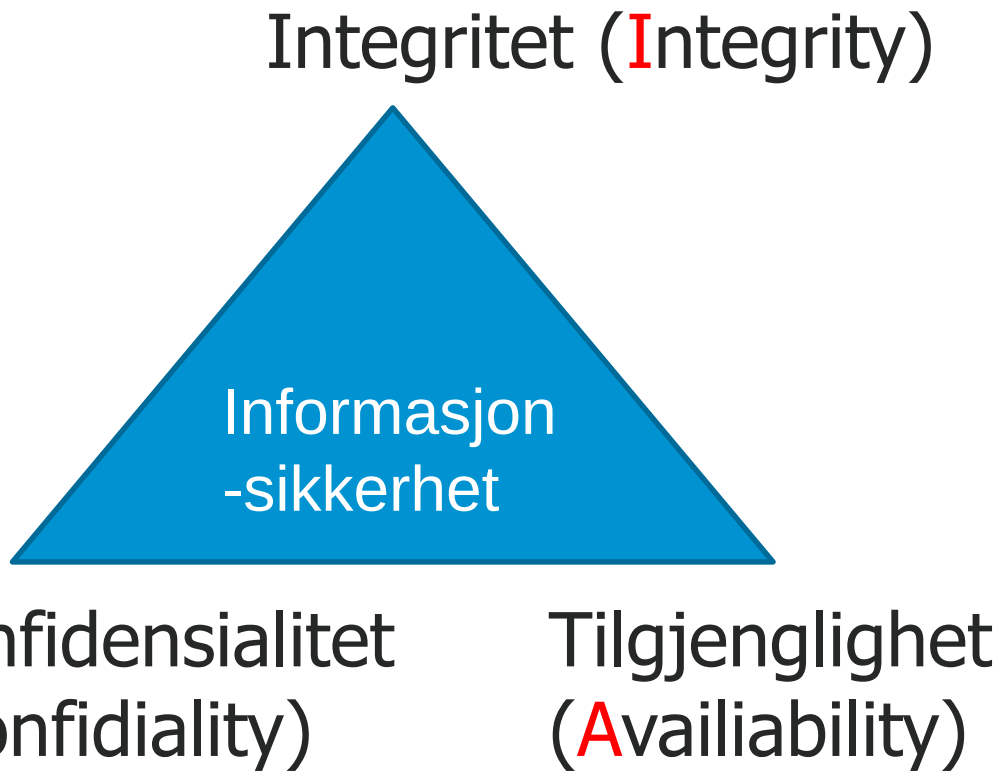
- Filadministratoren i OS'et må:
  - Lagre informasjon sikkert på utstyr (lagringsenhet: harddisk, …)
  - Kartlegge **sammenhengen** mellom **blokklagring** på utstyret og **filabstraksjonen** (logisk "bilde")
  - Allokere/deallokere lagringsenheter
  - Administrere deling av data
  - Beskytte mot feil, crash og ondsinnede brukere
  - Samtidskontroll (lock)



•Filabstraksjon

NIVÅER for sikkerhet og tiltak

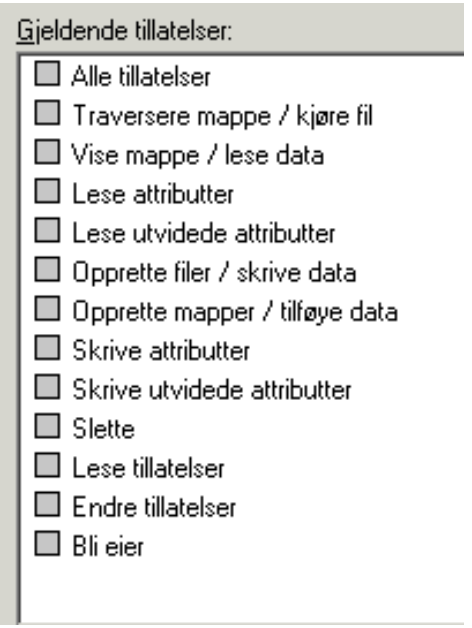
- Data
  - Access Control Lists, kryptering (av filsystem), …
- Applikasjon
  - Patching, sertifikater, Anti-virus, …
- Vertsmaskin
  - OS, autentisering og autorisering, brukeradm, group policies ….
- LAN (Internt nett)
  - IP-segmentering, IPSec, IDS, …
- Grense (Perimeter)
  - IDS, Firewall, VPN, NAT, …
- Fysisk sikring
  - Vakt, låser og overvåking
- Policy, prosedyrer, bevissthet
  - Brukeropplæring



Sikkerhet og filsystemet

- Både Linux/OSX og Windows benytter ulike typer ACL (**Access Control Lister**)
- Linux filsystemene deler adgangsrettigheter (execute, read, write) ut fra grupperingen
  - user
  - group
  - all
  - Kataloger (d) er bare en spesiell type bruker
  - Kun brukeren root har alle rettigheter/er administrator

- Windows (NTFS) har et mer finmasket og komplisert system
  - Vanlige brukere er ofte satt opp som administratorer også




```
-rw-r--r-- 1 blistog users    256 Sep 15  2008 TUBE.COM
-rw-r--r-- 1 blistog users 148480 Sep 11  2008 UTF-8AA.jpg
drwxr-xr-x 3 blistog users   4096 Nov 18  2008 xhtml
```

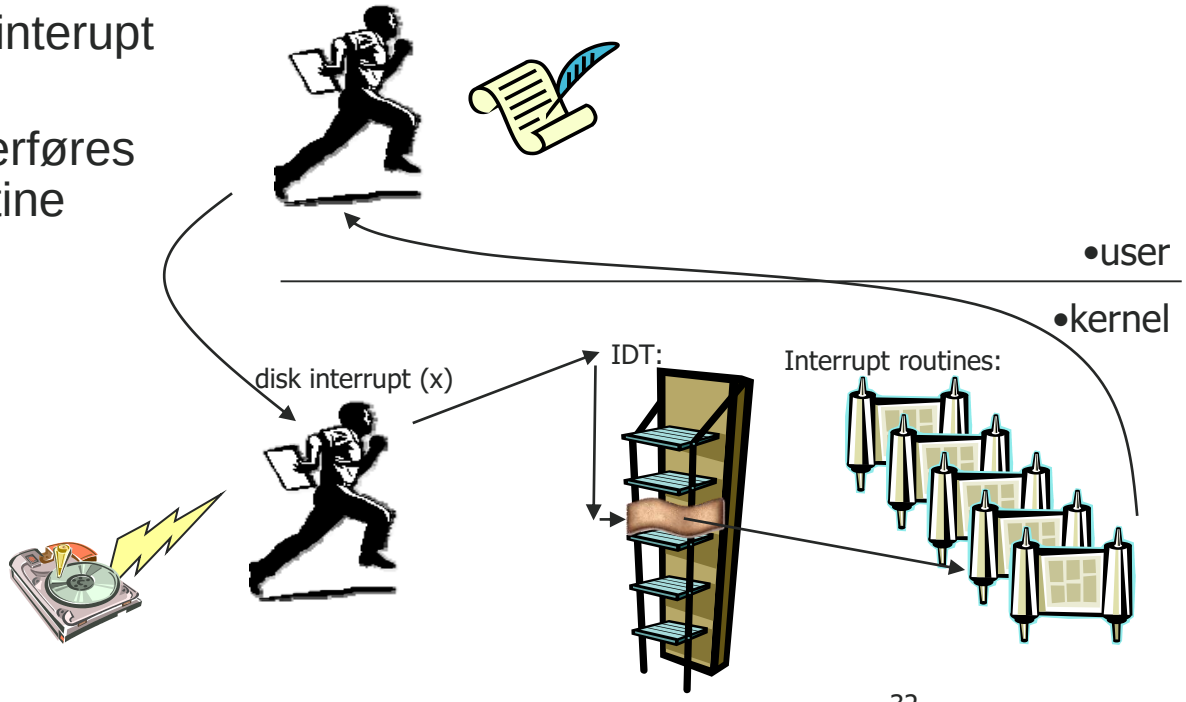
Operativsystemet – I/O kontroll

- Alt utstyr som er koblet til datamaskinen oppfattes som ressurser av operativsystemet
  - Skjerm, tastatur, mus, CD, høytalere, ...
- Operativsystemet må vite hvordan datamaskinen skal kommunisere med alt utstyret (drivere)
  - Plug and play
- Operativsystemet skal presentere utstyret på en enkel og grei måte, også for programmering
  - **UNIX/Linux**: ”Lat som” **alt er filer** som kan leses/skrives mot. (lastbare moduler)
  - **Windows**: ”Lat som” **alt er objekter**. (komplisert objekt-rom, mange nivåer av drivere)

Interrupt (og Exception) Håndtering

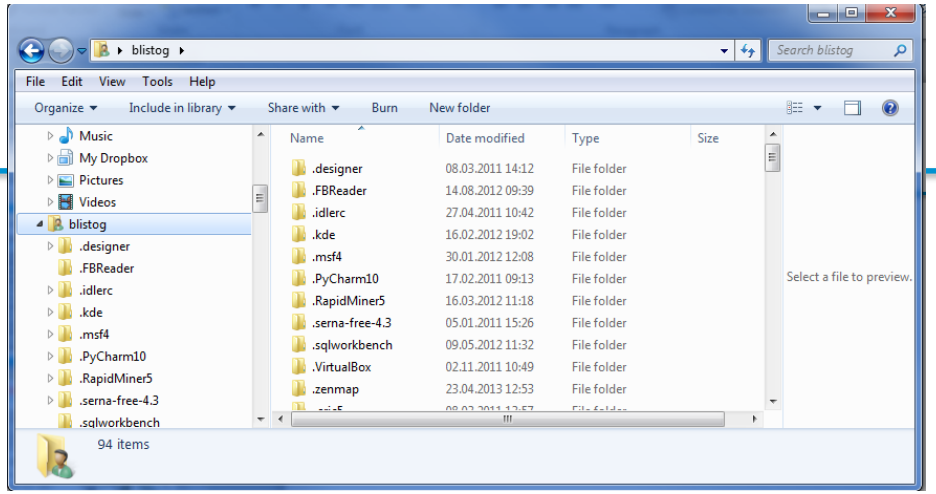
- IA-32 arkitekturen har en IDT med pekere til 256 ulike interrupt and exceptions
  - 32 (0 - 31) forhåndsdefinert og reservert
  - 224 (32 - 255) bruker/OS-definert
- Hvert interrupt har en en peker i Interrupt tabellen (IDT) og en unik index verdi som gir håndtering som følger

1. prosess kjører, det kommer et interupt
2. bevar kjøretilstand, kontroll overføres og finn interrupt-håndteringsrutine
3. Eksekver interrupt-håndtering
4. hent tilbake avbrutt prosess
5. fortsett eksekvering



Drivere

- Drivere er programvare som kan oversette frem og tilbake mellom OS sine krav og instruksjonsettet til kontrolleren på utstyret.
- Hovedårsaken til krasj er dårlig skrevne drivere
  - Microsoft påstår ”minst” 70%



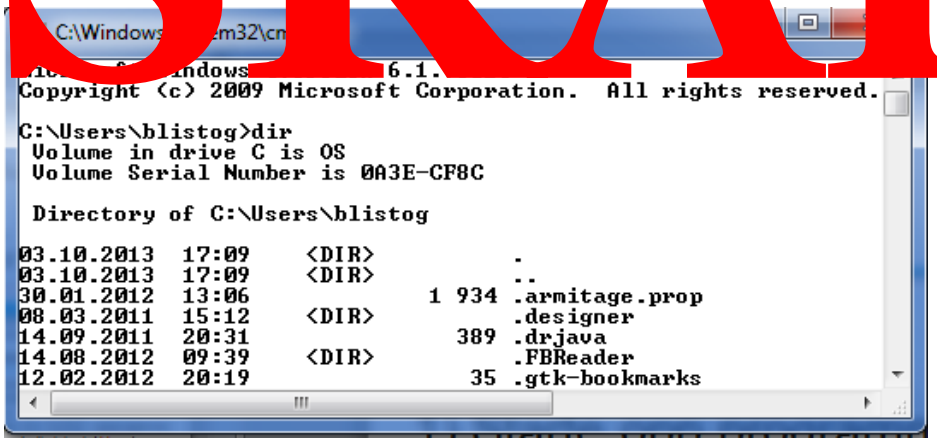
Viktigst i vanlig bruk:

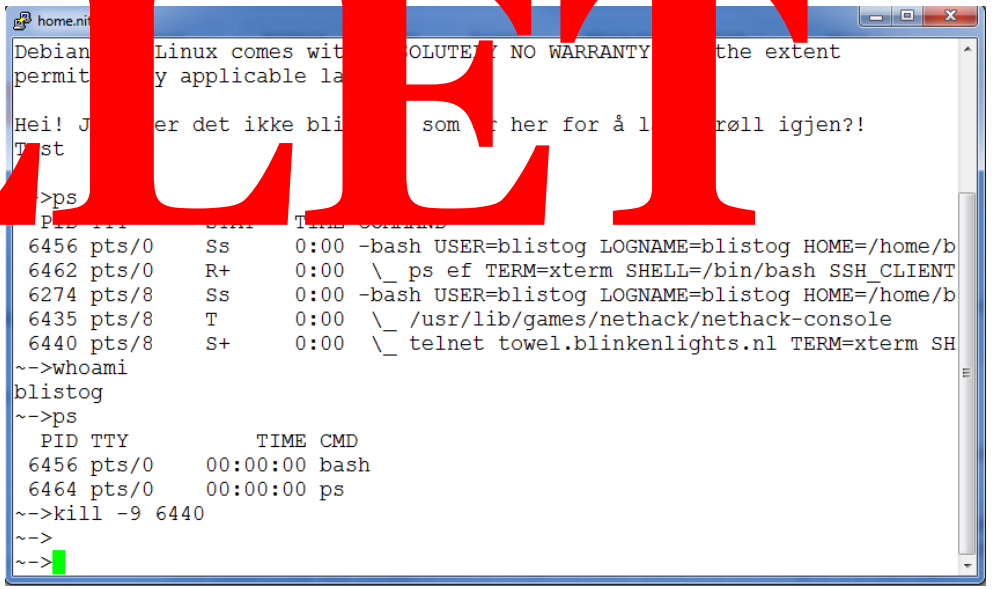
1) Starte opp programmer («prosesser»)

2) Manøvrere seg rundt i filsystemet

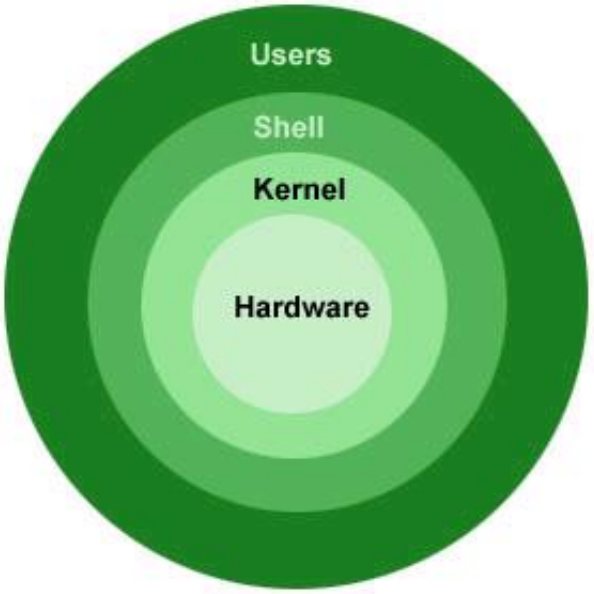
BRUKERPERSPEKTIV:

SKALLET





Brukerperspektiv: Skallet



- Brukeren kan gi ordre til OS via skallet (shell)
  - Starte opp programmer
  - Flytte filer
  - …
- Windows
  - Explorer (GUI)
  - Cmd.exe (NT)
  - Powershell (neste..)
- Linux
  - F.eks. Nautilus (Gnome)
  - bash (m.fl.)
- OS X
  - Finder (GUI)
  - bash (m.fl.)

”DOS” (**cmd**.exe) – UNIX/OSX (**bash**)

- Likheter
  - Tekst- og kommando-basert
  - Hierarkisk filstruktur
  - Tilsvarende fil-adgang (les, skriv, ...)
  - Standard I/O med piping (>, >>, <, |)

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights res

C:\Users\blistog>dir
 Volume in drive C is OS
 Volume Serial Number is 0A3E-CF8C

 Directory of C:\Users\blistog

14.09.2011  20:31    <DIR>          .
14.09.2011  20:31    <DIR>          ..
08.04.2011  11:28    <DIR>          .android
08.03.2011  15:12    <DIR>          .designer
14.09.2011  20:31               389 .drjava
27.04.2011  10:42    <DIR>          .idlerc
21.03.2011  16:29    <DIR>          .IntelliJIdea10
15.03.2011  13:40               192 .jline-jython.history
17.02.2011  10:13    <DIR>          .PyCharm10
```

```
blistog@nih-stud-web02:~
blistog@home.nith.no's password:
Last login: Mon Sep 20 22:53:28 2010 from nith-vpn-nat02.osl.basefarm.n

Hei! Jøss, er det ikke blistog, som er her for å lage krøll igjen?!


~->ls -la
total 29480
drwxr-xr-x   28 blistog users     4096 Jul 30 01:30 .
drwxr-xr-x 1298 root    root     36864 Sep 23 12:01 ..
-rwxr-xr-x    1 blistog users        0 Sep 26  2009 a.out
-rw-r--r--    1 blistog users        4 Oct  6  2008 arg
-rw-------    1 blistog users    10412 Sep 20 23:20 .bash_history
-rw-r--r--    1 blistog users     6126 Oct  9  2008 bash_igjen
-rw-------    1 blistog users     4934 Oct  9  2008 bash_igjen.save
-rw-r--r--    1 blistog users       17 Feb  6  2008 .bash_login
-rw-r--r--    1 blistog users       24 Sep  9  2003 .bash_logout
-rw-r--r--    1 blistog users      334 Feb  5  2008 .bash_profile
-rw-r--r--    1 blistog users      141 Feb  6  2008 .bashrc
```

Kommandoer CMD vs BASH

	DOS	UNIX
• display list of files	`dir`	`ls`
• display contents of file	`type`	`cat`
• copy file	`copy`	`cp`
• rename file	`ren`	`mv`
• delete file	`del`	`rm`
• delete directory	`rd`	`rmdir`
• find string	`find`	`grep`
• create directory	`md`	`mkdir`
• change working directory	`cd`	`cd`
• get help	`help`	`man`
• print file	`print`	`lpr`
• Endre rettigheter	`attrib`	`chmod`

Kommandoer og flagg

- Hvordan en kommando oppfører seg kan modifiseres ved å sette ulike flagg
  - plasseres direkte eller etter en flaggmarkør (*/*, **-**, **--**)

```
dir /?
ls -la
ps aux
```

-

Kommandoer og omdirigering

- Vanligvis er output til skjerm (konsoll), men i tillegg har du noen standard omdirigeringsoperatorer

>

  - Skriver output inn i en fil
  - Ex `dir > katalog.txt`, eller
    `ls -la > directorylist`

>>

  - Skriver output inn på **slutten** av en fil

<

  - Tar **innholdet i en fil** og bruker det som argumenter til kommandoen

|

  - Tar output fra en kommando og leverer den som input til neste kommando
  - Ex: `dir | sort /R | more`
    `ps aux | grep blistog | wc`
  - **«pipe»**

Signaler

- Under både Windows og Linux/OSX kan du i CLI-skall («command line interface» = kommandolinje-grensesnitt) gi signaler til OS med noen hurtigtaster
  - `Ctrl-C` Aborter kjørende prosess
  - `Ctrl-D` Slutt på filen (EOF)
  - `Ctrl-Z` Sett kjørende prosess til å sove
  - m.fl.

Hvorfor bruke kommando-skall?

- GUI er utmerket til «peke, dra, klikke» oppgaver
  - Dårlig til f.eks. automatisering pga **fattig semantikk**
  - Kommando-skall tilbyr oftest et fullstendig programmeringspråk
- Servere
  - De fleste web-servere kjører Apache på Linux, og settes ikke opp med GUI da det ville være dårlig ressursbruk.
  - Microsoft har også begynt å satse mer på skall enn GUI for servere.. (**Powershell**)

Hardware har styrt hva OS tilbyr/gjør;

Ny hardware har ofte kommet til ut fra OS behov.

HISTORIKK

Operativsystemet - tidlig utvikling

- Frem mot 1960-tallet var det ikke bruk for operativsystem. Datamaskinen ble brukt av en person eller en jobb om gangen
- Ved batch-kjøring måtte maskinen ha en kø-ordner
- Ved time-sharing (multitasking) måtte maskinen holde rede på flere prosesser på en gang
- Bruk av masselager krevde et organisert filsystem
- Virtuelt minne krevde bedre styring mot filsystemet
- Stadig nytt eksternt utstyr krevde forenklet I/O-kontroll
- Brukere var datafolk

Operativsystemet - videre utvikling

- Komplisert oppstart-prosedyre
- Flere interaktive brukere på samme maskin
- Behov for generelle service-programmer for brukerne
- Maskiner i nettverk
- Beskytte maskiner og brukere mot maskiner og brukere

- PCer gjennomgikk samme utvikling som større maskiner allerede hadde
- Økende tilbud av applikasjoner på maskin og nett
- Tyngden av brukerne uten databakgrunn

Spesialiserte operativsystemer

- Parallelle systemer
  - Mer enn en CPU i samme maskinen
  - Kan kjøre flere jobber samtidig
- Distribuerte systemer
  - Flere maskiner jobber sammen
  - Deling av jobber og ressurser
- Sanntidsystemer
  - Tidskritisk responstid, styrt av interrupt
- Nettverk

- Mobiltelefoner o.l.

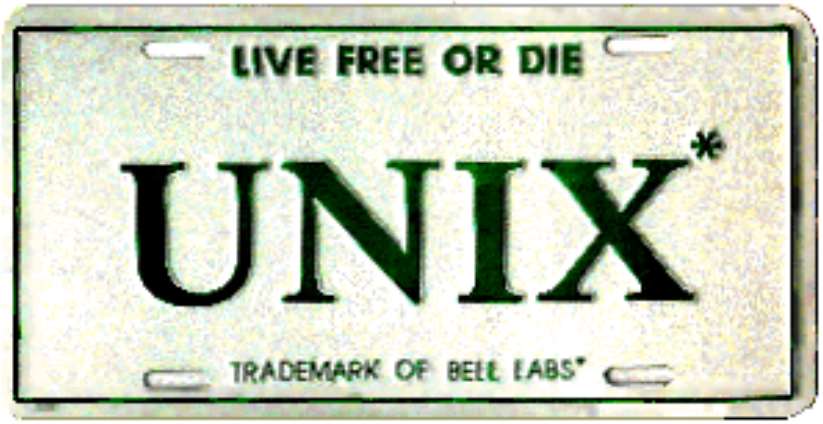
Operativsystemer

- UNIX
  - Skrevet i C (1971), tekstbasert, portabelt, flerbruker
  - Linux som moderne (PC) utgave
  - Stor frihetsgrad både fordel og ulempe
- MS-DOS
  - Skrevet i maskinkode (1980) av Microsoft for IBM
  - Tekstbasert, enbruker
  - Senere bygget Windows 3.0-11 GUI oppnå DOS
  - Liten frihetsgrad

DOS

- DOS hentet mye fra UNIX og kan derfor synes å være relativt likt
  - CP/M: Gary Kildall, 1973 for 8080 og 8” diskett (DR-DOS)
  - 86-DOS: Tim Paterson, 1980 for 8086
    - **FAT** filsystem
  - MS-DOS: Microsoft for IBM, 1981, basert på 86-DOS som de kjøpte for 75.000 dollar
- Hoved-forskjellene ligger ikke så mye i kommandoene men i frihet, struktur og portabilitet

DOS - UNIX



- Forskjeller
  - DOS var en-bruker, UNIX var fler-bruker
  - DOS støttet bare enkel-prosessering, UNIX støtter multi-prosessering
  - DOS hadde kommando-tolker (command.com -> cmd.exe), UNIX bruker skall (shell, bash, csh, tcsh,...)
  - DOS/NT kjøres på Intel-kompatible prosessorer, UNIX er mye mer portabel
  - DOS/NT brukte bat og cmd-filer, UNIX bruker skall-skript

UNIX historikk

- Startet som Multics på MIT 1965
  - Første time-sharing OS
- Videreutviklet hos Bell 1965-1969
  - Ken Thompson, Dennis Ritchie,.....
- Overført fra PDP-7 til PDP-11 1970
  - Unics → Unix
  - Skrevet i C
  - Dannet basis for portabilitet
- Linux - Linus Torvalds 1991

Forenklet tidskart!

MULTI-TASKING

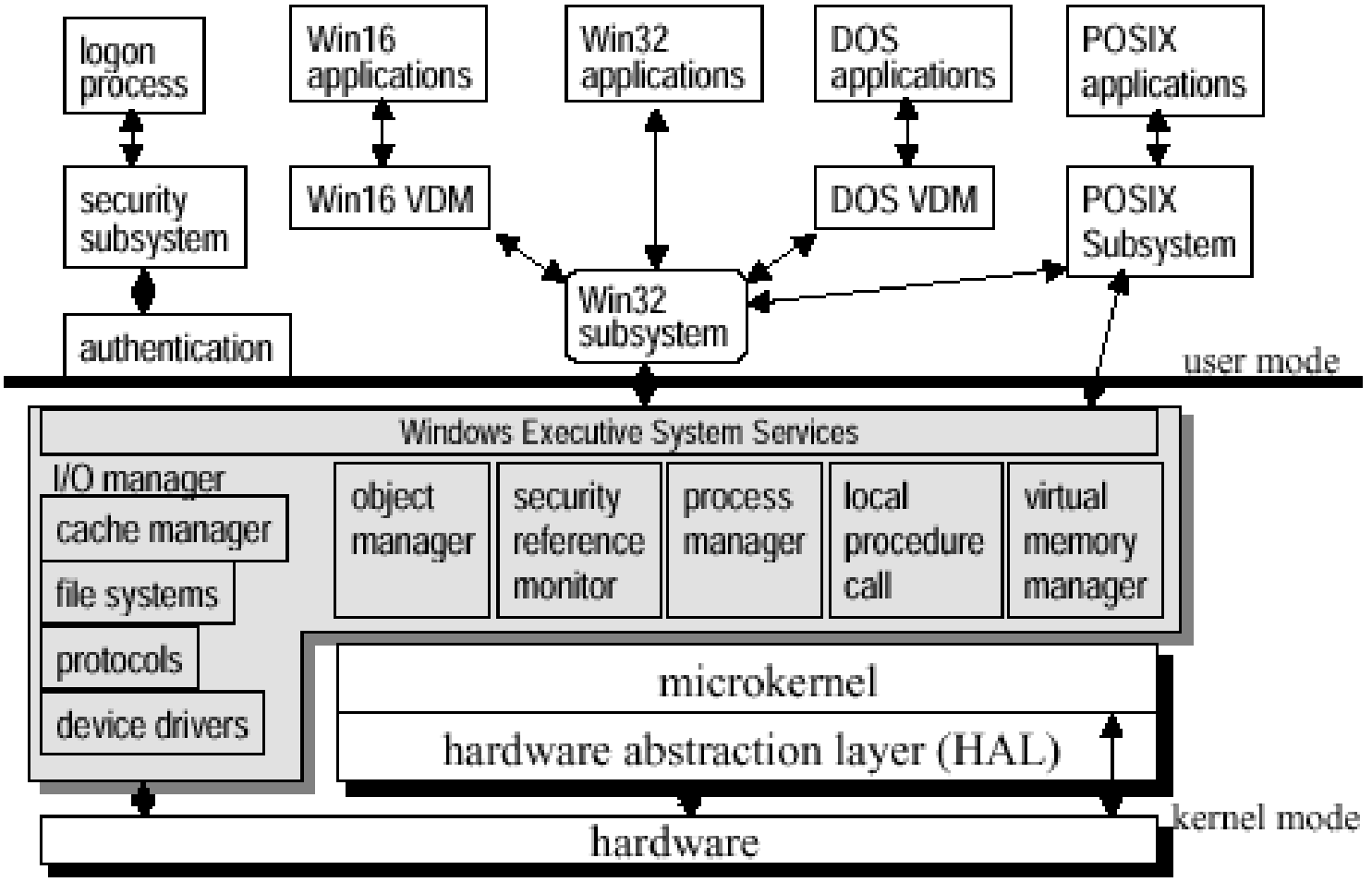
- Alle moderne OS tilbyr multitasking
- Flere prosesser kjører «samtidig»,
  - «samtidig» betyr at de får tildelt tidsrom på CPU for å kjøre sin(e) tråd(er)
    - eller kjøres på hver sin prosessorkjerne
    - Synkroniserings-problemer
  - OS besørger kjøreplan («scheduling»)
  - OS sørger for minne-isolasjon og –samarbeid mellom prosesser/tråder

Microsoft Windows

- Windows 1.0, 2.0, 3.0 og 3.1 er grafiske grensesnitt for DOS
- Windows 95/98 inneholder DOS som en service
- Windows NT var en mer portabel utgave av Windows
  - Nytt kjerne-design fra bunnen av. Ikke basert på DOS
  - Modulært, 32 bit system for mange samtidige omgivelser
  - Robuste mekanismer for kritiske prosesser
  - Kjører alle Windows-applikasjoner med samme interface
- I Windows XP møttes Windows 98 og Windows 2000
- Vista tilfører ny sikkerhets-modell og grensesnitt
- Windows 7 og Server 2008 er fortsatt samme kjerne (NT)
- Windows 8 og 10 er neste trinn, men involverer ingen/små endringer i kjernen (Windows Vista er internt «Windows 6.0», Windows 10 er internt «Windows 6.3»…)

Windows

- System struktur



Windows

- Registry
  - Sentralisert database for system konfigurering informasjon
  - Videreføring av DOS og Windows INI-filer
  - Brukes av forskjellige Win enheter
    - Recognizer: Gjenkjenner hardware ved oppstart
    - Setup: Verktøy for system konfigurering
    - Kernel: CPU konfigurering og balansering av system belastning
    - Drivers: Informasjon om initialisering av hardware

Windows

- Registry
  - Hierarkisk oppbygning
    - HKEY_ LOCAL_MACHINE
    - HKEY_ CLASSES_ROOT
    - HKEY_ CURRENT_USER
    - HKEY_ USERS
    - HKEY_ CURRENT_CONFIG
    - HKEY_DYN_DATA
  - Oppdateringer valideres ikke mot andre deler av registry
  - Backup og restore er følsomt

```
REGEDIT4

[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\explorer\Advanced\Folder\StartMenuScrollPrograms]
"Type"="checkbox"
"Text"="Multi-Column Start Menu"
"HKeyRoot"=dword:80000001
"RegPath"="Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Advanced"
"ValueName"="StartMenuScrollPrograms"
"CheckedValue"=dword:00000000
"UncheckedValue"=dword:00000001
"DefaultValue"=dword:00000000
"HelpID"="update.hlp"
```

Windows

- NT File System (NTFS)
  - Stor forbedring i forhold til DOS's FAT
  - Bedre egenskaper og ytelse
    - Data gjenvinning, sikkerhet, store filer, komprimering
  - Metadata lagret i Master File Table (MFT)
    - Data om data: Hvordan, når, av hvem og formatering
  - NTFS er ikke bare byte-strømmer
    - MFT inneholder opplysninger om dataene
  - Filreferanser er mer enn bare adresser
    - 48 bit filnummer (peker), 16 bit sekvensnummer (multidisk)
  - Logging av transaksjoner
  - NTFS videreutvikles ikke og venter på en arvtager

OSX (forhistorie)

- Apple startet også med et «rent» DOS (Disk Operating System) 1976-1984
  - Ikke multitask, 1-nivå filsystem («uten kataloger»)
- MacIntosh introdusert i 1984
  - Finder som GUI-«skall»
  - Nytt filsystem etter hvert : Hierarchical File System
  - Toolbox for GUI-app utvikling

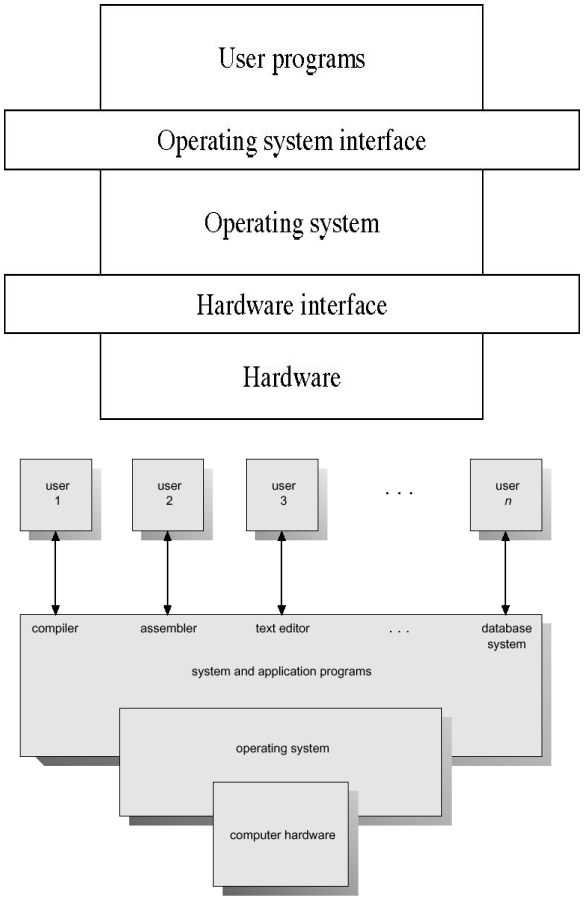
OSX

- Videreutvikling av NeXTSTEP
  - Basert på **Mach mikrokjerne**, FreeBSD (Unix) og I/O Kit driver-pakken
  - **Kjernen** kalles Darwin/**XNU** (X is Not Unix)
    - Mye Open Source takket være FreeBSD bakgrunnen

OPPSUMMERING

Operativsystemet – generelle krav

- Operativsystemet skal altså organisere de tilgjengelige ressursene og administrere dem på best mulig måte både for maskiner og brukere
- Brukerne vil konsentrere seg om applikasjoner uten å tenke på maskinenes software- og hardware- finurligheter
- Brukerne vil helst bli presentert for et miljø som er uavhengig av foreliggende hardware
- Opplærings-terskelen skal helst være så lav som mulig



Hva skal vi kunne?

- Definere prosess, tråd og ressurs
- Forklare hvordan OS abstraherer Hardware og dermed tilbyr et enklere programmerings- og bruker-grensesnitt
- Kunne forklare OS rolle som ressurs-administrator: Prosess/tråd-adm, Minne-adm, Fil-adm, Utstyrs-adm.
- Vite hvilke problemer som må løses for å få til multitasking
- Kjenne minne-layouten og kjøremønsteret til et vanlig program på en Intel-type prosessor

Hva skal vi kunne?

- Forklare hva et interrupt er og sammenhengen med I/O
- Forklare forskjellen på User- og Kernel-modus på CPU
- Forklare hva Virtuelt minne går ut på

Prate om eksamen

- Første eksamen er en ny opplevelse for de fleste, mange har kanskje lært mye om hvordan eksamen er (noen på den harde måten)
- Hva skjer hvis jeg får en F?
  - Du må ta opp faget som konte eksamen
  - Å klare å ta eksamen som konte hvis du stopper å komme i forelesningene nå vil bli veldig vanskelig
  - Mange fag bygger på det vi skal lære frem mot jul i dette faget, så hvis du stopper å komme i forelesningene ødelegger du også for andre fag!
- Når får jeg vite karakteren min?
- Etter 17 kan dere velge om dere vil gå til øving, eller om dere vil bli igjen i Auditoriet en liten stund hvor jeg vil gå gjennom eksamensoppgavene og prate litt om de