Hvorfor har vi operativsystemer?

- Portable programmer, enklere å forholde seg til.
- Abstraherer bort maskinvaredetaljer. Hvordan lagre filer? Hvordan sende data til nettverkskort. Vi som bruker diverse enheter vet ikke hva som skjer bak kulissene ettersom operativsystemet gjør det for oss.
- Deling av ressurser (scheduling, allokering). Rettferdig fordeling mellom prosesser
- Sikkerhet. Prosesser skal ikke kunne lese hverandres minneområder.

Hvilke to perspektiver bruker vi for hva er et operativsystemer er?

- Extended machine (utvidet maskin): Gjemmer detaljene slik at den blir enklere å bruke
- Resource manager (Ressurshåndterer): Deler ressursene mellom programmene.

Hvorfor trenger vi å vite hvordan operativsystemet fungerer?

- For å kunne bruke maskinen på en effektiv måte. Kjøretid og latens er viktige mål på effektivitet
- For å forstå begrensninger, muligheter og mulige feil. Helst før de oppstår.

Hvilke hovedkomponenter finner man i et operativsystem?

- Filhåndtering
- Brukergrensesnitt (system calls)
    - Dvs funksjoner som kjøres fra en gitt applikasjon.
    - Når brukere interagerer med input-enheter som musepeker og tastatur, så vil applikasjonen kallet på gitte system call() for å koble opp mot operativsystemet som videre kobler opp mot hardware enheter
- Prosesshåndtering
    - Vil si hvordan vi håndterer prosesser som kjører et gitt program de har å eksekvere. Dette med at ikke alle prosesser kan utnytte cpu´en samtidig, men avhengig av måten “scheduler” er satt opp
- Enhetshåndtering
- Kommunikasjon (IPC, Nettverk)
- Minnehåndtering (allokering, aksess)

Hva skjer når vi kaller en systemfunksjon (f.eks read) fra brukernivå (user space)?

- 1. Argumentene pushes på stacken.
- 2. Systemkallnummer legges i et register.
- 3. Exceptioninstruksjonen kalt en “trap”.
- 4.Operativsystemet håndterer systemkallet.
- 5. Operativsystemet returnerer, og programmet kjører videre.

Hva er forskjellen mellom “Interrups” og “Exceptions”? Hva er et systemkall?

- Asynkron vil si at man ikke har en tidsrelasjon med CPU´en. Synkron vil si at man har en direkte kobling med CPU´en.
- Interrupts er asynkrone. Ikke trigget direkte av prosessor-instruksjonen. Dvs at ikke alle tastetrykk ved tastatur, vil bli trigget av prosessoren. Men det er som oftest periferenhetene som mus, tastatur, mikrofon og videoer som knyttes opp mot interrupts.
- Exceptions er synkrone. Trigget av prosessor-instruksjon.
- Systemkall er synkron, og derfor en exception. Et systemkall kalles av og til en trap

Hva gjør bootstrap?

- Bootstrap setter i gang oppstartsprosedyren for operativsystemet. Bootstrap laster inn og kjører bootprogrammet fra en kjent lokasjon på disk. Kan sees som at man “Booter” opp eller “starter opp alt.

Hvilke to typiske kjerne-arkitekturer har vi? Hva er forskjellen?

Disse er monolittiske og mikrokjerner som er eksempler på kjerne-arkitekturer(OS). Forskjellen mellom disse er følgende:

- **Monolittiske kjerner** er kjerner hvor alt av funksjoner er satt inn i kjernen. Dette er effektivt, men er komplisert på grunn av oppbyggingen av en slik kjerne(mange funksjoner satt sammen) og ved feil under kjøring så kan kjernen krasje.
- **Mikrokjerner** derimot har oppbyggingen slik at hovedfunksjonen er plassert inn i mikrokjernen. Videre er mikrokjernen koblet opp mot hardware-komponentene som er blitt vist over. Fordeler ved arkitekturen er at mikrokjerner er små, utvidbare, enkelt å modulere og porterbar. Ulempen er at håndtering av systemkall kan ta lang tid ettersom vi må aksessere diverse hardware-komponenter som må passere meldinger frem og tilbake.