2.2 The evolution of operating systems

Tuesday, March 31, 2020 12:20 AM

Seriell prosessering

Betegner operasjonen med scheduling (tidsstyring) og setup time – den gamle måten programmene kjørte på før OS ble introdusert. Veldig tungvint. Navnet reflekteres gjennom hvordan brukere har tilgang til computeren i serier.

Enkle batchsystemer (Uniprogrammert)

Tidlige computere var dyre, og det var derfor viktig å maksimere utnyttelsen av prosessor. For å forbedre denne utnyttelsen ble det innført et konsept i form av et batch OS. Den grunnleggende ideen bak et batch OS var en del av software kalt monitoren. Med en slik type OS hadde ikke lenger sluttbruker direkte aksess til prosessoren. Hvert program er konstruert til å forgreine tilbake til monitoren når den fullfører prosessen, hvorpå monitoren automatisk starter å laste neste program.

For å forstå dette bedre kan man se på perspektivet til henholdsvis monitor og prosessor:

Monitor: Monitoren kontrollerer sekvensen av handlinger. For at dette skal fungere må monitoren ligge i hovedminnet og være tilgjengelig for execution. Denne delen er referert til som *resident monitor.*

Prosessor: På et tidspunkt utfører prosessoren instruksjoner fra den delen av hovedminnet som inneholder monitoren. Disse instruksjonene gjør at neste jobb (jobb, *def: et enkelt program*) blir lest inn i en annen del av hovedminnet. Når en jobb har blitt lest inn, vil prosessoren bli instruert om å fortsette execution av bruker-programmet av monitoren. Deretter vil prosessoren fortsette å kjøre bruker-programmet til den er ferdig eller det oppstår en error. Begge disse tilfellene gjør uansett at prosessoren fetcher sin neste instruksjon fra monitor-programmet.

Et batch OS, eller monitoren, er egentlig et enkelt computer program. Den avhenger av egenskapen til prosessoren å fetche instruksjoner fra ulike steder i hovedminnet for å veksle mellom å ta og gi kontroll.

Likevel hadde man større behov for:

- Minnebeskyttelse: ikke tukle med den delen av minne som inneholder monitoren mens den kjører brukerprogrammet.
- En *timer*
- Priviligerte instruksjoner: noen instruksjoner er priviligerte i form av at de bare kan utføres av monitoren
- Avbrudssmekanismer

Vurderingene av minnebeskyttelse og priviligerte instruksjoner skapte et konsept om en dualitet av moduser. Brukerprogrammer blir utført i brukermodus, hvor noen deler av minnet er beskyttet fra brukerens bruk, og hvor noen instruksjoner ikke kan bli utført. På den andre siden; systemmodus eller kjernemodus (*kernel mode*), der priviligerte instruksjoner kan utføres og hvor beskyttede deler av minnet kan aksesseres.

Kort fortalt om uniprogrammert batch OS:

Prosessortiden alternerer mellom execution av brukerprogrammer og execution av monitoren. Dette går på bekostning av: hovedminnet – der deler blir gitt til monitoren, og noe av prosessortiden går til monitoren. Til tross for dette forbedrer uniprogrammert batch systemutnyttelsen av computeren.

Multiprogrammerende batch systemer

Som forklart tidligere så vil en prosessor i utgangspunktet måtte vente og forbli inaktiv mens en I/O-enhet gjør seg ferdig. Med denne ovennente modellen må det hvert fall være plass til OS-et (resident monitor) og ett brukerprogram i hovedminnet. Multiprogrammerende batch systemer utnytter at vi kan utvide minnet til å holde enda flere programmer, slik at vi kan bytte mellom hvilket program som kjører. Dette er hensiktsmessig når en prosessor venter på en I/O-enhet, for da kan man enkelt bytte til en annen prosessor som kjører et program som mest sannsynlig ikke benytter seg av I/O. Dette er kalt multiprogrammering, eller multitasking.

Multiprogrammerende OS-er er mer avanserte sammenlignet med uniprogrammerende OS-er. For at de skal kunne ha flere jobber klare til å kjøre, så må de befinne seg i hovedminnet, og krever dermed en viss grad av minnehåndtering.

Tid-delende systemer (time-sharing systems)

På samme måte som multiprogrammerende batch systemer tillater prosessoren å håndtere flere batch jobber på en gang, så kan de også håndtere flere interaktive jobber. Denne teknikken kalles tid-deling,

ettersom prosessortiden er delt mellom flere brukere. I et tid-delende system er det flere brukere som aksesserer samtidig gjennom terminaler, hvorpå OS-et utfører hvert brukerprogram i en kort snutt eller en kvantum av utregning.

Time slicing er en teknikk der OS gjenopptar kontroll og tildeler prosessor til en annen bruker ved hvert klokkeavbrudd.

Table 2.3 Batch Multiprogramming versus Time Sharing

	Batch Multiprogramming	**Time Sharing**
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Selv om tid-deling og multiprogrammerende systemer kan oppfylle grunnleggende behov så er det nye problemer de innfører for et OS: hvis flere jobber ligger i minnet, så må de beskyttes fra å forstyrre hverandre, som for eksempel ved å endre hverandres data. Med flere interaktive brukere må filsystemet beskyttes så bare autentiserte brukere har tilgang til visse filer.