DIS - Skalerbarhed 1 - Operativsystemer

Processer i kernen / fork()

fork()
Der er et systemkald der hedder fork() - et systemkald til at kopiere processer. Kerner bruger det hele tiden når programmet skal startes.

En proces er en instant af noget, der bliver afviklet
Det betyder, at når vi starter et program, så kører vi en fil der indeholder noget maskinkode - instruktion til processoren om at gøre et eller andet. Det bliver gjort og hukommelsen bliver aktiveret, og når det sker, oprettes en instans af programmet.

En proces har en mængde ressourcer til rådighed og har mulighed for at afvikle tråde og andre processer --> derfor det hedder forking. Når en proces starter en anden proces i en computer, så sker der forkning. Når vi starter et program (word), sker det via et andet program (Finder fx) - et operativsystem fx. Det er forking.

Så er der skrevet.Skype/ for at starte programmet - så starter terminalen Skype - og fordi det køres sådan, kan man se at Skype kører beskeder tilbage på den måde. Der findes noget, der hedder Standard-in, standard-out og standard-error - terminalprocessen er her brugt til at starte Skype, den starter (forker) Skype. Skype er derfor vores fork i terminalen, og når vi kører skype på denne her måde, kan vi se, hvad den sender tilbage til den oprigtige proces, som startede denne proces.

Copy on Write
sørger for at fork() *ikke bruger hukommelse*, før det er nødvendigt

Når man forker noget, så den bliver proces, der bliver lavet, kopierer man sine forældres omgivelser. Copy-on-Write sørger for at man ikke bruger hukommelse, medmindre den proces der starter, ændrer sig.

Hvad består OS af?

Kernel

Hvad er et operativsystem? Uanset hvilket OS, så har det en kerne - det er her det hele foregår. Operativsystemer har kerner (ikke det samme som CPU kerner, der er noget andet).

Kernen --> en slags koordinator / vicevært, der sørger for at programmer kan få adgang til processoren, at det kan gemmes data i hukommelsen og så videre.

Når man afvikler mange programmer, er det vigtigt at holde styr dem - virtual adressering (nok ikke noget vi bliver spurgt ind til til eksamen), det er noget som kernen tilbyder programmer, som gør at man ikke overskriver hinandens programmer i hukommelsen. Hukommelsen er en lang række af blokke, hvor man kan putte ting ind - overskrider man disse blokke (fx ved hacking, overflows), så kan man få programmet til at gå ned - fordi de data, som programmet troede der var adgang til, var der ikke alligevel.

Hvad er en kerne? En fil som ligger på computeren rent fysisk som ligger på harddisken. Når jeg tænder min mac bruger den denne fil til at starte med.

Den må ikke slettes, fordi så får vi en lang eftermiddag. Kernen er et stykke software, som er kompileret - den lavet om til maskinkode, som ikke går i stykker, medmindre placeringen af denne på harddisken er i stykker, men så er hele kernen korrupt.

Hvis man er hardwareproducent som fx Apple, så har man en fordel ved at man selv sidder og bestemmer hvilket hardware der sidder i. Så kan man lave kerner, der er skræddersyet til hardware og kun den hardware. Kernen er tailored til kun at understøtte den hardware, der er i computeren --> det er derfor, Mac starter hurtigt op! Helt klart en kæmpe fordel.

Vi har nogle **applikationer** som snakker med

Som taler med **enheder** - CPU, RAM, Enheder. kan være tastatur, mus, skærm, batteri

kernen som ligger under de her applikationer, som sådan set er operativsystemer

Tråde / Threads

En tråd er et redskab, som en proces kan bruge til at afvikle en instruktion. En tråd deler hukommelsesområde med sin proces og er afhængig af selvsamme.
En proces har altid som minimum én tråd. En tråd kan fx indeholder en instruktion om at dividere to tal, som kernen så kan udføre - resultatet kommer derefter tilbage
Skal man snakke om rigtig multithreading = her snakker man om flere kerner - flere tråde (udregninger) kan afvikles på flere forskellige kerner

Man bruger ofte en semafor til at låse en proces, så et program ikke kan startes flere gange. Det heder også et PID (Process ID) knyttet til en specifik proces.
Inde i terminal kan man bruge det til at lukke et program: "kill -9 *procesID*" – så lukker det program, som har det specifikke ID.
Semafor = en logisk nøgle man ligger et sted

Virtualisering

Virtualisering er en form for distribuering af et system, der giver mulighed for afvikling af enten delvist afhængige eller uafhængige miljøer.

Udnyttelsen af ressourcer på en computer - altså CPU, RAM og så videre - er ofte ikke effektiv over tid, hvorfor det giver mening at virtualisere på den ene eller anden facon.

Virtualisering er en form for distribuering af et system, der giver mulighed for at køre flere operativsystemer på den samme server. Det kan køre flere OS på samme server på den samme maskine

Parallels, docker, VirtualBox, VmWARE

Distribueret Schedulering / Algoritmer

Afvikling af noget, der er scheduleret

Når man har en række maskiner (servere, alt muligt andet) og sætter dem sammen, så har man koordineret hvor man siger "den her computer laver noget", "den her computere laver ikke en skid", når man så skal koordinere opgaver, kan man sende dem ud til en specifik maskine.

Resource management component of a system which moves jobs around the processors to balance load and maximize overall performance. – Typically makes sense in LAN level distributed systems due to latency concerns

First-in-First-Out
den mest simple form for scheduling, hvor instruktioner afvikles i den rækkefølge, som de modtages i.

Shortest-Remaining-Time
afvikler instruktioner, der tager kortest tid, først. Det kan resultere i starvation af længerevarende processer

Round-Robin
giver alle instruktioner en del af den samlede CPU- tid. Hvis en instruktion ikke når at blive færdig indenfor tidsrammen, så går turen videre til den næste. Næsten altid denne der tales om

Distribueret schedulering - faktum at man udnytter en server bedre, det blev brugt hos Twitter sammen med noget der hedder Mesophere. Twitter kunne spare så mange servere gennem distribueret schedulering, fordi de installerede mesophere - som gjorde at CPU tiden blev mindre - de sparede en masse

ressourcer og omkostninger på dette - meget smart

Netværksbaserede OS

Applikationer
applikationer (safari, outlook)

Operativsystem
(macOS, Windows)

Hardware

Google Go(pher)

Netværksbaserede og fuldt ud distribuerede - sidstnævnte er en **pipe dream**.
Netværksbaserede = alle ressourcer ligger tilgængelig på netværk -

Der findes OS som er netværksbaserede - systemer der konsumerer baseret på netværk og så er der OS som er baseret på fuldt ud distribuerede systemer. Det gør der ikke heltrigtigt, vi har ikke fundet ud af at lave det endnu - Google Go er åbent programmeringssprog fx er tæt på - er det tætteste vi kommer på fuldt ud distribueret er der, hvor alle ressourcer ligger tilgængelige på en netværksbaseret struktur - fx Plan 9 af Rob Pike (canadisk programmør,

fantastisk og gudsbenådet) - eksempel på DIS der er fuldt ud distribueret.