Professionshøjskolen UCN
IT-Uddannelserne
Datamatiker
Gruppe: 2
Normalsider: 3

Vejledere: Karsten Jeppesen

Jesper Kvejborg Lemvig, Mathias Vilsgaard Rasmussen, Martin Lentz Skøttrup, Anders Lund Kjeld Hansen, Thomas Petersen

Indholdsfortegnelse

Indledning

Windows er et styresystem som er benyttet af en stor procentdel af verdenens computere. Dette styresystem har haft flere forskellige versioner igennem de sidste mange årtier. Den nyeste version er nu Windows 10, som denne rapport omhandler. Rapporten dækker om Windows 10's arkitektur, process og thread model, memory management og scheduleing og til slut i/o model.

Windows 10

Arkitektur

Windows 10 findes i flere arkitekturer, de mest udbredte er 64-bit kompatible men det findes også med 32-bit. Forskellen på 64-, og 32-bit er mængden af hukommelse som computeren er i stand til at benytte sig af. 64-bit modellerne er kompatible med op til 6 TB RAM, hvor 32-bit er låst til et maksimum på 4 GB RAM.[1] Dette lyder som en stor forskel og det er det givetvis også, men da det kun er "Windows 10 Pro for Workstations" som kan benytte sig af op til 6 TB RAM og de resterende styresystemer er låst til deres egen maksimale udnyttelse af RAM er dette også en potentiel faktor når man vælger styresystem. Det er dog tydeligt at se at disse konfigurationer er lavet efter bestemte målgrupper, som de også er navngivet efter.

De forskellige Windows 10 arkitekturer er "Windows 10 Home", "Windows 10 S", "Windows 10 Pro", "Windows 10 Pro for Workstations", "Windows 10 Education", "Windows 10 Pro Education", "Windows 10 Mobile", "Windows 10 Enterprise", "Window 10 Mobile Enterprise", "Windows 10 Enterprise LTSC", "Windows 10X".[2] Disse forskellige versioner af styresystemet gør det muligt at tilpasse det til et stort udvalg af brugere og giver brugere mulighed for at vælge en konfigurering der passer til dem. Bl.a. er sidste nævnte "Windows 10X" lavet til "specifikt" hardware i den forstand at det blev skabt for at understøtte flerskærmsenheder, i form af at understøtte bestemte positioner og gestures herpå, men nu er udviklet til at kunne understøtte flerskærmsenheder samt konventionelle enkelt enkelt-skærms enheder.[3]

[1] https://www.groovypost.com/howto/windows-10-edition-architecture-build-your-install/
[2] https://computing.which.co.uk/hc/en-gb/articles/115004450029-Windows-10-which-version-is-right-for-you-
[3] https://en.wikipedia.org/wiki/Windows_10_editions

Process og thread model

I Windows er en process de forskellige softwarer som kører, disse processer har hver især et nummer som identificerer dem, dette kaldes et ID. En thread er her så et objekt som bestemmer hvilken del af programmet kører og disse threads har også et ID som identificerer dem fra hinanden. Formålet med en thread er at allokere processortid og en enkel process kan altså have mere end én thread, hvilket vil sige at der altså kan bruges flere threads på en process. På en maskine som kun har en enkel processor kan flere threads blive allokeret ad gangen, men kun en thread kan køre ad gangen. Det virker på den måde at en thread kører i et kort stykke tid hvorefter udførelsen af opgaven bliver sendt videre til den næste thread. Hvis der er flere processorer i en maskine, giver det mulighed for at multi-threading kan foregå, dette giver en fordel hvis en applikation eller software har flere threads, da man her kan køre de forskellige threads samtidigt på forskellige processorer.

Windows kernel-mode behandler udførelsen af alle threads som kører i en process. Ligegyldigt om man har en eller flere processorer skal man stadig være forsigtig indenfor driver-programmering og være sikker på at alle threads i en process er designet så de virker ordentligt ligegyldigt rækkefølgen de bliver udført i.

Memory management og scheduling

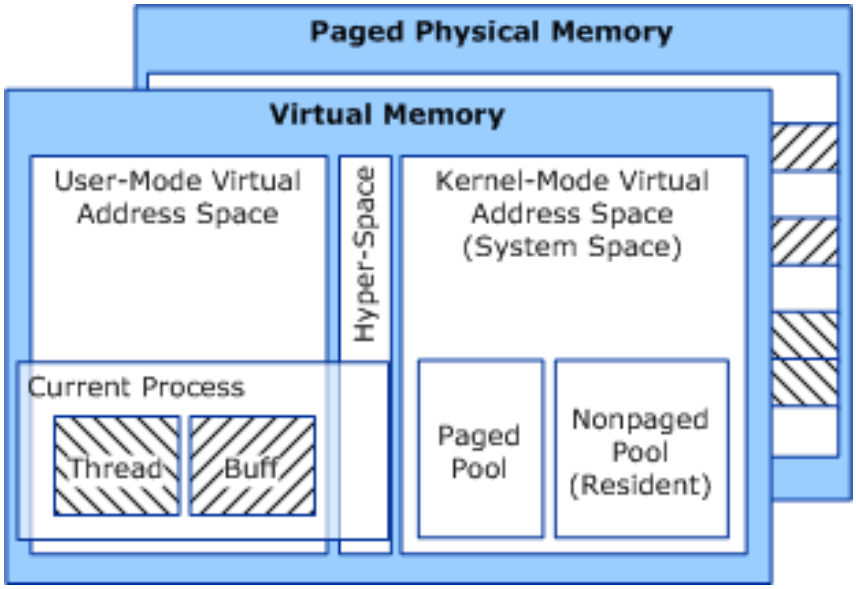
I Windows er det "Kernel-Mode Memory Manager" der styrer fysisk hukommelse for operativsystemet. Hukommelse som primært er i form af RAM (Random Access Memory). Memory Manager allokerer også hukommelse til opgaver, som f.eks. opbevaring af interne data, buffering data under i/o operationer, og deler hukommelsen med andre Kernal-mode og User-mode komponenter. Windows styrer virtuel og fysisk hukommelse, og deler hukommelsen ind i separate bruger og systemadresserum. En driver kan selv specificere om den tildelte hukommelse understøtter evner, som f.eks. demand paging, data caching, og instruction execution.[4]

Når der bliver gjort brug af demand paging, bliver pages kun loadede når de bliver refereret til. [5]

En cache fungerer som en form for mellemlager, som bliver brugt til midlertidige beregninger, før de f.eks. bliver gemt på en harddisk. [6]

Der er tre trin i en normal instruction execution cyklus. Første trin vil være at instruktionerne bliver hentet fra hukommelsen. Andet trin er at de instruktionerne bliver decoded af decoderen. Tredje trin er at instruktionerne bliver udført. [7]

[4] https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/managing-memory-for-drivers
[5] https://en.wikipedia.org/wiki/Memory_paging
[6] https://da.wikipedia.org/wiki/Cache
[7] https://en.wikipedia.org/wiki/Instruction_cycle



Billedet her giver et godt overblik over Windows Memory Space. Det kan ses hvordan den virtuelle hukommelse bliver understøttet af paged fysisk hukommelse. User-Space virtuel hukommelse og System-Space hukommelse allokeret fra en paged pool vil altid være pageable. Alt der har med User-Space code eller data at gøre kan til enhver tid blive paged ud til sekundær lagring, selv mens processen allerede er i gang. [8]

I/O model

Generelt har alle OS en i/o model for hvordan data fra knap så relevante devices skal flow fra eller til dem. En af de gode ting som Windows i/o model har, er dets support af asynkront i/o flow.

Windows Kernel-mode i/o manager er det som styrer kommunikationen imellem applikationer og interfaces som vores devices (i form af f.eks. mus, disk drives, video controllers… osv.) drivers giver os. Det er ikke altid tilfældet at vores operativsystem og vores devices arbejder i samme hastighed, derfor foregår kommunikationen imellem disse to elementer hovedsageligt med i/o request packages, som også er kendt som IRPs. Disse requests bliver sendt ud fra operativsystemet til de individuelle drivers og derefter videre ud til andre.
Generelt benytter windows i/o sig af en layered driver model kendt som stacks. Vores IRPs går igennem en driver til en anden for at danne kommunikation. Dette vil altså sige, at hvis man forbinder en mus til computeren, så vil driveren kommunikere igennem USB hubben, som derefter skal kommunikere videre til USB host controlleren, som til slut skal kommunikere til resten af computeren igennem en PCI bus.
Det er påkrævet at når IRPs bliver sendt afsted, så bliver de også lavet inden for forholdsvis kort tid, da ellers er der stor sandsynlighed fort at det crasher. Dette sker, da hvis en driver som er en del af stacken ikke enten modtager, håndtere eller videre sender informationer, så kan det resultere i crashes.

[8] https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/overview-of-windows-memory-space

i/o manageren har to underkomponenter i form af ”plug and play manager” og ”power manager”. De håndterer funktionaliteten bag de to. Plug and Play er det som gør at vi kan f.eks. smide et usb-stick i vores computer, og så bliver det tilføjet til filsystemet og er klar til brug.
Power Manager er det i systemet som gør, at computeren ved hvor meget power den kan give og skal give til individuelle komponenter for at de samt systemet køre optimalt som en helhed.

Ligesom selve operativsystemet, så er i/o driversene object-baseret. Dette vil altså sige, at drivers, deres devices og alt form for system hardware bliver vist og bearbejdet som objekter. Så når noget skal gøres, så eksportere i/o manageren nogle kernel-mode routiner som driverne kan kalde på for at få deres arbejde gjort ved, at manipulere forskellige objekter.

Ordliste

Ord	
OS	Operativsystem
i/o	Input / output
IRPs	i/o request packages
RAM	Random Access Memory
LTSC	Long-Term Servicing Channel

Kilder / Sources

https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/