Oppsumeringshefte OS-del IN2140

Hva er et OS?

- Et OS er en samling verktøy og programmer som danner et grensesnitt mellom en sluttbruker og maskinvare. OS håndterer grunnleggende oppgaver slik som minnehåndtering, fil systemet, I/O kontroll og håndtering av prosesser.

Hva er et monolittisk OS?

- Et monolittisk OS er et operativt system som er strukturert slik at hele operativsystemet kjører i kernel-space. Drivere legges til OS'et i form av moduler. Tjenester slik som fil-system, minne, IPC, kjører i samme space.

- Monolittiske OS er mer utsatt for system-feil, siden blant annen drivere kjører i kernel-space. En driver feil vil utsette hele systemet for en crash.

Hva er en mikrokjerne OS?

- Et mikrokjerne-OS er et OS som er strukturert slik at tjenester (fil, IPC, minne) kjører i kernel space, men tjenester slik som drivere, fil servere kjører som user-space prosesser. De fleste operativsystemet i dag er ikke mikro-kjerner.

- Mikrokjerner er typisk bedre sikret mot system-feil som følge av dårlige drivere

- Mikrokjerner er typisk tregere enn monolittiske, på grunn av mer kompliserte mekanismer som må være på plass for å kommunisere mellom prosesser

Hva er en hybrid-kjerne OS?

- En hybrid-kjerne er et OS som tar i bruk både mikro-kjerne paradigmet og det monolittiske. Et slikt OS er strukturert slik at visse prosesser kjører som user-space og får tilgang til kernel-space ressurser gjennom et grensesnitt. Kernel-space kan igjen ha et grensesnitt mellom seg og maskinvaren.

- De fleste operativsystemer i dag er hybrid-kjerner

Hva er «user-space?»

- User-space er et spesifikt nivå i et operativ-system. Prosesser som kjører i dette nivået har begrenset med ressurser, instruksjoner og tilganger. Programmer som du starter som bruker, vil havne her.

- Typisk kalles dette også for ring 3.
- Minne som vi får tilgang til tilbys som virtuelt minne. Vi får illusjonen å ha et gigantisk adresserom, når vi faktisk ikke har det.

Hva er «kernel-space?»

- Kernel-space er et spesifikt nivå i et operativ-system. Prosesser som kjører her, har få begrensninger. De kan typisk ta i bruk alle ressurser, instruksjoner og har tilgang til alt av minne.

- Typisk kaller vi dette ring 0.

- Prosesser som kjører her er blant annen drivere, lyd-systemet, schedulering, og minne håndtering.

Hva er «ring-modellen?»

- Ring-modellen er en modell som visualiserer ulike nivåer av privilegier i operativ-systemet. Den ligner veldig mye på en løk og ser slik ut.

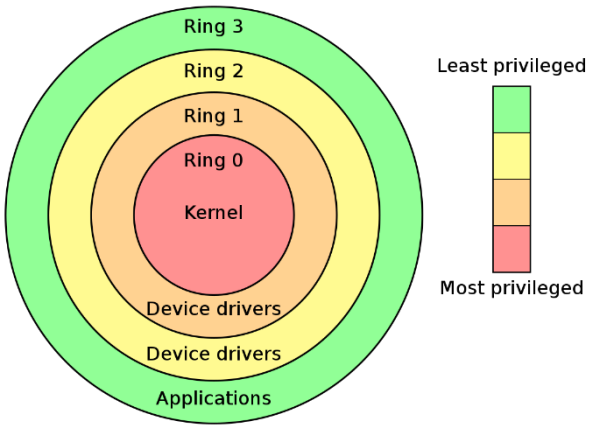

Figure 1
https://commons.wikimedia.org/wiki/File:Priv_rings.svg

- Jo lavere tall, jo mer privilegert er vi. Merk at det også finnes -1 og -2. Disse brukes når OS'et kjører i en viritualiserings-tjeneste.

Hva er en prosess?

- En prosess er en kjørende instans av et program. Prosesser har prosess-nummer, minne-område, og en tilstand. En prosess blir eksekveres av en eller flere tråder.

- En prosess kan starte en ny prosess ved hjelp av et system-kall

- En prosess deler ikke minne med en annen prosess.

Hva er en tråd?

- En tråd er en eksekverings-enhet som tilhører en prosess.

- En prosess kan ha flere tråder, en eksekverende prosess må ha minst én tråd.'

- Tråder deler minne-området med hverandre.

- Tråder deler ikke blant annet program-teller, registere og stack.

Hva er en scheduler?

- En scheduler er en prosess som kjører i kernel-space som har i oppgave i å organisere en kø for alle prosessene som ønsker å få tid på CPU 'en.

- En scheduler kan anvende en scheduleringsalgoritme for å skreddersy hvordan denne køen oppfører seg.

Hva er en scheduleringsalgoritme?

- En scheduleringsalgoritme er en algoritme som scheduler prosesser i en viss rekkefølge basert på visse kriterier.

Eksempler på scheduleringsalgoritmer.

- FIFO Scheduleringsalgoritme.
    - Her er det den første prosessen som ankommer som får tid på CPU 'en. Etter at den er ferdig, blir den sendt tilbake til køen.
- SJTF
    - Her er det den oppgaven som har minst tid igjen til den er ferdig, som før kjøre.
- Round-robin
    - Her får alle prosesser en lik tids-enhet som de har lov til å kjøre, før de sendes tilbake til køen.
- Lottery
    - Her får alle prosessene en lotto-billett. Hver tids-enhet trekker schedulering et tall. Den prosessen som har det tallet, får lov til å kjøre.

Hva er en ikke-preemptiv scheduler?

- En ikke preemptiv scheduler, er en scheduler som ikke kan avbryte en prosess som kjører, med mindre prosessen frivillig gir fra seg tiden eller avslutter.

Hva er en «time-slice?»

- En time-slice er en tids-enhet som brukes i schedulering. Hver prosess får en time-slice og når den er ferdig, vil prosessen settes i slutten av schedulerings-køen

Hva er en preemptiv scheduler?

- En preemptiv scheduler er en scheduler som kan stoppe en prosess mens den kjører, for å så scheduler den senere.

Eksempler på preemptive algoritmer

- Round-Robin.
    - Her må hver prosess avsluttes av schedulering etter at de har fått sin time-slice.
- Shortest remaining time first
    - Her får en prosess CPU-tid, helt til en ny prosess med mindre tid igjen kommer inn i køen

Hva er en «dispatcher?»

- En dispatcher er den enheten som faktisk flytter en prosess fra scheduleringskøen, og til prossessoren. Prossessen forandrer tilstand når dette skjer.

Hva slags «tilstander» kan en prosess ha?

- Ready
    - En prosess er klar for å scheduleres.
- Running
    - En prosess har CPU-tid akkurat nå.
- Terminated
    - En prosess har avsluttet. Den kan ikke scheduleres.
- Waiting
    - Prossessen venter i scheduleringskøen
- Blocking
    - Prossessen venter på ressurser fra I/O.

Hva er «minne?»

- Minne er hvor vi lagrer data i datamaskinen.

Hva er primær-minne?

- Primærminne er typisk det vi kaller RAM (Random Access Memory). Denne typen minne er veldig raskt, men ikke veldig stort.

- Primær-minne er volatilt. Det vil si at vi mister alt som ligger her når vi slår av maskinen, mister strøm, kræsjer osv.

Hva er sekundær-minne?

- Sekundær-minne er minne som vi skriver til når vi ønsker å lagre minne.

- Typisk er dette hard-disker (HDD) og SSD-er (Solid state drive)

-
- Sekundær-minne er ikke volatilt, men har betydelig tregere å skrive/lese fra. Det er enda tregere å lese/skrive til slik minne hvis vi hopper mye fram og tilbake. (non-sequential read/write)

- Slik minne er typisk mye større.

Hva er eksternt minne?

- Eksternt minne er minne som typisk brukes for arkivering av data.

- Eksempler på dette er tape, trommel minne og minne-pinner

- Slik minne har den største kapasiteten, men den verste lese/skrive hastigheten (unntak til begge punktene, minne-pinner)

Hva er «cache»

- Cache er minne internt inne i CPU'en.

- Cache-minne er mye raskere enn primær-minne, men har veldig lite kapasitet.

- Cache er delt opp i flere nivåer. Typisk fra L1-L3. L1 er det raskeste, men med minst kapasitet.

- I de fleste multicore arkitekturer, er L3 cachen delt mellom flere kjerner, mens hver kjerne har en egen L1 og L2 cache.

- Størrelser kan være fra 4-32Mb L3 cache, ned til 1Kb til 4Kb L1 cache.

Hva er adressering av minne?

- Adressering av minne er hvordan vi får tilgang til spesifikt minne som vi ønsker.

- Hvordan vi adresserer minne avhenger av adresserings-teknikken.

    - Direkte adressering (absolute adressing)
        - Leser med å adressere bytes direkte (f.eks. 0x00000000 eller 0xffffffff)
        - Veldig raskt.
    - Relativ adressering
        - Adressering relativt til et punkt i minne (f.eks. når vi adresserer fra et array, adresserer vi *relativt* fra base-adressen.)

Hva er «swapping?»

- Swapping er en teknikk hvor vi bytter ut minne som ikke er brukt ofte, med minne som er brukt veldig hyppig.

- Typisk swapper vi hele prosesser, ved å skrive tilstanden til disk, for å så skrive den til minne når vi trenger det.

Hva er «paging?»

- Paging er en teknikk som lar oss lagre deler av minne på disk, for å så bytte de tilbake igjen til primært minne.

- I paging har vi bedre kontroll over hva vi skriver til disk, vi kan skrive minne-områder til disk.

- Paging er mer komplisert å implementere, i motsetning til swapping, som swapper hele prosesser, mens paging lagrer deler av minne.

- Pages er alltid lik størrelse.

Hva er «page-faults?»

- Page faults er feil som oppstår når vi forsøker å få tak i en page, men av diverse grunner ikke greier det.

Hva slags «page-faults» kan vi få?

- Vi har tre store kategorier av page faults.
    - Minor
        - Her er en page lastet inn i minne, men ikke markert at den er lastet inn og klar for bruk.
    - Major
        - Her er en page ikke lastet inn i minne, og må lastes inn, før den kan brukes
    - Invalid
        - Her er det en prosess som forsøker å få tak i en page som den ikke har lov til å få tak til (OS-minne, read-minne, null-pointers o.l), eller som er utenfor adresserommet. Disse vil typisk gi segmentation-fault

Hva er en «page-replacement algoritme?»

- En page replacement algoritme er en algoritme som setter opp regler på når en page skal hentes og når den skal kastes ut.

Hva slags page-replacement algoritmer har vi?

- Her er noen algoritmer og hvordan de funker.
    - FIFO (First in first out)
        - Første page som kommer inn, er den første page som skal ut når vi må hente inn en ny page.
    - Second chance
        - Når en page er lastet inn, og en forespørsel om denne pagen kommer inn, setter vi den bakerst i køen, og later som den er en helt ny page.
    - Clock
        - Vi anvender en ring-buffer lignende struktur, og vi har en teller (hånd) som peker på den siste etterspurte pagen. Hvis pagen har blitt nylig etterspurt, så bytter vi ut pagen med en ny en, og beveger «hånden» et hakk videre. Hvis pagen er ny, så gir vi den en ny «sjanse» og beveger hånden til en page som har blitt etterspurt minst en gang.
    - Least recently used (LRU)
        - Vi har en liste av pages, hvor de mest etterspurte er på starten, mens de minst etterspurte er på slutten. Når vi har en page fault, bytter vi ut den minst etterspurte.

Hva er «virtuelt-minne»

- Virtuelt minne er minne som tilsynelatende eksisterer for prosesser, men i realiteten så er dette minne ikke fysisk minne.

- Hver prosess ser tilsynelatende 4GiB med minne på en f.eks. 32-bit maskin, mens i realiteten eksisterer det kun 2GiB med fysisk minne.

- Oppnås ved hjelp av paging mekanismer og en pagefil på maskinen. Når fysisk minne fylles opp, skrives pages til fil, når vi får en page fault, skriver vi de tilbake.

Hva er en «MMU»

- En MMU (Memory Management Unit) er en brikke typisk inne i CPU'en som håndterer oversetting av adresser i virtuelt minne, til fysiske adresser.

- En MMU spør en annen komponent, en TLB. Hvis adressen er hyppig aksessert, vil den ligge i TLB'en. TLB'en er en assosiativ cache.

Hva er en «TLB»

- En TLB (Kort for translation lookaside buffer). Er et buffer som tar vare på kartlegginger mellom virtuelle addressser og fysiske addresser.

- TLB'en er typisk oppfører seg som en assosiativ cache, hvor vi bruker den virtuelle adressen som en nøkkel for en verdi.

- TLB'en reduserer tidsbruket med å oversette addresser, gitt at de er i selve TLB'en.

Hva er en «hard-disk?»

- En hard-disk er typisk fellesbegrepet for minne-lagringsmedium som er et sett med plater som deretter leses av et lesehode. Platene er laget av magnetiske materialer, slik som neodynium

- Slike disker er vesentlig tregere enn alternative ikke-volatile lagringsmedier, slik som solid state drives.

- Når vi aksesserer data, så må lesehodet finne riktig spor og lese. Har vi mye tilfeldig aksess, vil ytelse være svært dårlig, siden lesehodet må bytte spor flere ganger.

Hva slags lagringsmedier er ikke-volatile?

- At et lagringsmedium er ikke-volatilt, betyr at vi kan koble fra strøm til disken, uten tap av data.
- Hard-disker og solid-state drives er eksempler på ikke-volatile lagringsmedium.

- RAM og CPU-cacher er volatile. Når de mister strøm, mister de data som ligger der.

Hva er en «solid-state drive»

- En solid state drive, er et lagringsmedium som tar i bruk lagringsbrikker som selve lagringsmedium.

- Brikkene er flash-minne. Slik minne «flashes» som skriver eller sletter minne. Flash-Prossessen forandrer verdiene på de logiske kretsene (NAND/NOR kretser) som forandrer verdien.

- Betydelig mye raskere på sekvensiell og tilfeldig aksess, typisk dyrere og av og til mindre kapasitet (for pris) enn hard-disker

Hva er en «SSHD»

- Kalles også for en hybrid-disk. Har plater for lagring av data, men bruker en solid-state cache for å forbedre aksesstid.

- Et mellom-valg mellom SSD og HDD.

Forskjeller mellom hard-disk og Solid state

- Hard-disker er:
    - Typisk billigere per gigabyte/prisenhet.
    - Typisk mer lagringsplass/prisenhet
    - Tåler ikke sterke eksterne magnetfelt (sletter data)
    - Betydelig tregere sekvensiell aksess av minne
    - Utrolig mye verre ikke-sekvensiell aksess av minne.
    - Få måter å koble disse på. (SATA-kabel, eSATA) som kan begrense aksesstid.
- Solid state drives
    - Typisk dyrere per gigabyte/prisenhet
    - Typisk mindre lagringsplass/prisenhet
    - Ikke påvirket av eksterne magnetfelt (med mindre de er ekstreme)

    - Bedre toleranse for støt.
    - Betydelig raskere aksess i sekvensiell og ikke-sekvensiell aksess.
    - Mange måter å koble slike drives. (SATA, eSATA, NVME PCIE)

Hva er en «disk-scheduleringsalgoritme»

- En disk-scheduleringsalgoritme er en algoritme som bestemmer hvordan vi beveger disk hodet.

Hva slags disk-scheduleringsalgoritmer har vi?

- FCFS (First come first serve)
    - Første forespørsel som kommer inn, er den som disk hodet beveger seg til.
- SSTF (Shortest seek time first)
    - Beveger disk hodet til den forespørselen som er nærmest disk hodet.
- SCAN
    - Fungerer som en heis. Hodet skanner fra side til side, og leser forespørsler som den skanner over.
- C-SCAN
    - Skanner fra en side til en annen. Når den er ferdig med siste forespørsel på kanten, beveger hodet til den andre kanten, og beveger seg innover på nytt og tar hånd om forespørsler.
- LOOK
    - Fungerer som og C-SCAN, men vil ikke bevege seg til ytterste spor med mindre det er en forespørsel der. Beveger seg til siste forespørsel og vil deretter bevege seg innover som SCAN og svare på forespørsler
- C-LOOK
    - Fungerer likt som C-SCAN. Vil bevege svare på forespørsler, helt til siste forespørsel. Etter siste, vil den bevege seg til innerste forespørsel, og begynne derifra igjen.

Hva er «IPC?»

- IPC står for inter-process communication

- En måte for prosesser å snakke med hverandre, sende data osv.

Hva er en «pipe»

- En pipe er en form for IPC from typisk er relatert til *nix lignende systemer (UNIX, BSD, MacOS). De er en måte for IPC, som lar oss sende output fra et program, gjennom en pipe, til input for et nytt program.

Hva er «shared-memory»

- Shared memory er en annen form for IPC.

- Her kartlegges minne mellom to eller flere prosesser. Prosesser kan skrive og lese fra dette minne og kan kommunisere

- Viktig! Skriving av samme minne må koordineres gjennom en mutex eller lignende mekanisme.

Hva er en «mutex?»

- En mutex er kort for «mutual-exclusion». En mutex forsikrer at det er kun én prosess som skriver til et minne området som er kritisk (kritisk: delt mellom flere prosesser)

- Mutex er en generalisering av en semaphore, hvor kun én prosess kan få tilgang til det kritiske området.

Hva er en «semaphore»

- En semaphore er en variabel/objekt som en prosess kan ta i bruk for å få tilgang til en kritisk seksjon. Semaphoren er synlig over alle prosessene, slik at man ikke kan feilaktig tro at man har lov til å skrive til minne, når en annen prosess har lov til det.

- En kritisk seksjon er minne/kode som kun én prosess skal få tilgang til om gangen.

Hva er et «signal»

- Signaler er en form for IPC i UNIX lignende systemer som programmer kan få. Disse ligner på interruptus, men kommer fra programvare.

- Mange signaler, noen av de kan håndteres.

Hva slags signaler har vi?

- Her er noen signaler
    - SIGHUP
        - Signal som kommer når prosessen som kontrollerer har hengt seg opp.
    - SIGINT
        - Signal som er en interrupt fra brukeren. Ctrl+C sender denne
    - SIGQUIT
        - Signal som stopper en prosess, sendes av bruker. Ctrl+D sender denne
    - SIGFPE
        - Udefinert matteoperasjon, f.eks. deling på null.
    - SIGKILL
        - Prossessen drepes umiddelbart. Sendes ved kill -9 <pid-nummer>
    - SIGALRM

        - Alarmer, kan lages med timers f.eks. timerfd
    - SIGTERM
        - Prosessen bes om å avslutte.

Hva er en «signal-handler?»

- En signal-handler håndterer signaler som gis til en prosess.

- De fleste signaler kan håndteres, mens SIGSTOP og SIGKILL kan ikke håndteres, og vil terminere prosessen.

- Signal-handlers kan implementeres for egne C programmer, slik at de kan håndtere de diverse signalene.

Hva er en fil-deskriptor?

- En fil-deskriptor er et tall som gis til en prosess når prosessen ønsker å ta i bruk ressurser som OS'et tilbyr gjennom et API.

- Hvis du åpner en fil/socket/timer o.l får du en slik fil-deskriptor.

- Du identifiserer ressursen med fil-deskriptor. Hver prosess har en egen slik tabell. Hvis du forker en prosess, deler de denne tabellen. To prosesser vil ikke ha samme tabell, og kan ha samme «tall» for en resurs.

Hva er «forking»

- Forking er å kalle på system-kallet «fork()».

- Fork starter en ny prosess, som er et «barn» av foreldreprossesen som kaller på fork.

- Retur-verdien til fork oppfører seg slik:
    - Foreldreprossesen vil få PID til barnet.
    - Barnet vil få «0» som PID.