Operativsystemet D-NIX

Den här beskrivningen ger en övergripande presentation av Diab Datas operativsystem D-NIX och den miljö som skapas med detta som grund. Vill du veta mer om de enskilda produkterna som ingår, se faktablad för aktuell delprodukt.

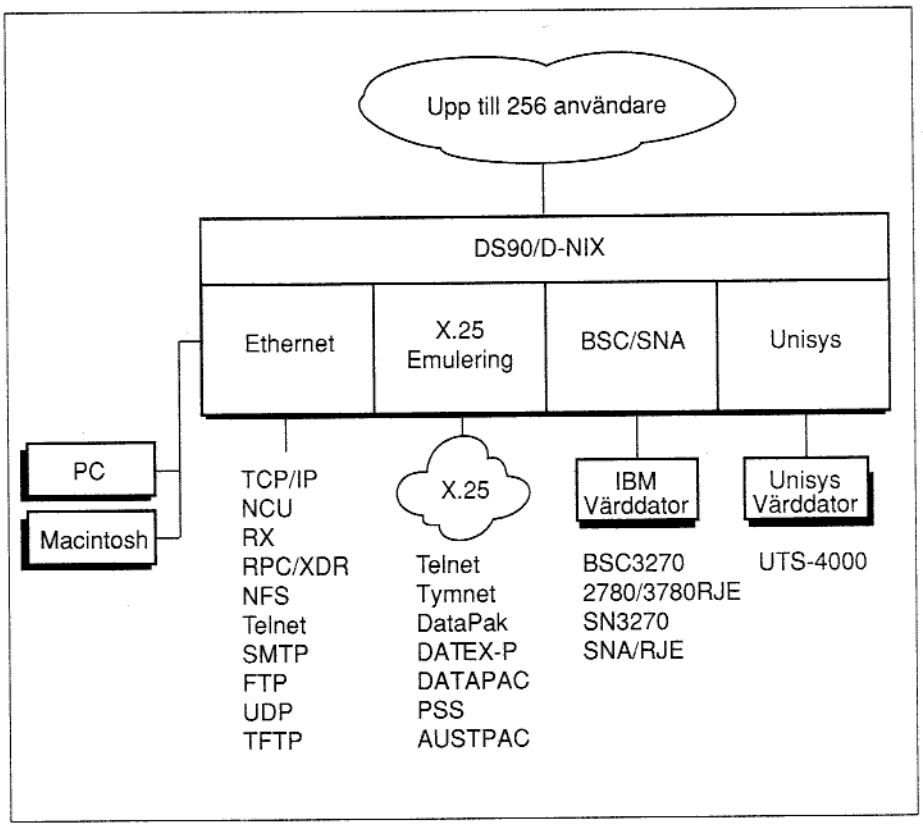
UNIX är ett av de populärare operativsystemen för mikroprocessorbaserade datorsystem. Fördelarna med UNIX är:

- Möjlighet till flera användare tack vare tidsdelning.
- Bra utveckligsmiljö med många kraftfulla och flexibla verktyg.
- Transparant hantering av olika enheter.
- Möjligheter till flyttbar kod.
- Många applikationer från tredjepartsleverantörer.

UNIX utvecklas till en standard som kommer att erbjuda basen för större likhet mellan system och flyttbarhet för användare. I dag används UNIX i system i storleksklasser från persondatorer till stordatorer. Diab Datas målsättning var att utveckla ett multiprocessorbaserat system med högklassig prestanda vad gäller maskin- och programvara. De kända begränsningarna i prestanda som finns i UNIX har man kommit över med förbättringarna i DS90-systemen (kombinationen maskinvara och D-NIX).

Detta häfte beskriver översiktligt D-NIX System 5.3. Målet är att ge en insikt i de unika D-NIX attributen och dess applikationer. Operativsystemet D-NIX introducerades 1983. D-NIX uppfyller kraven i SVID och POSIX (D-NIX uppfyller SVID för bassystemet och de utvidgade nivåerna i kärnan) och dessutom skrevs det från grunden för att stödja realtid och användningen av flera processorer.

Med kombinationen DS90 och D-NIX får användaren en mycket flexibel miljö och stora möjligheter att kommunicera med andra datorsystem.



D-NIX är ett portabelt operativsystem som för tillfället används på Motorola 680X0 processorer. Koden till D-NIX är huvudsakligen skriven i C. Eftersom D-NIX utvecklades med tanke på flyttbarhet kan systemet enkelt flyttas till andra CISC- eller RISC-baserade system med en eller flera processorer.

D-NIX erbjuder UNIX standardfunktionalitet plus realtid, transaktionshantering, OLTP (On-Line Transaction Processing) samt andra prestandahöjande funktioner. D-NIX 5.3 ger förutsägbara svarstider på användarinitierade händelser. Här följer en kort beskrivning av utökningarna i D-NIX:

- Stöd för flera processorer.
- Prioritetsbaserad tilldelning av processor.
- Tömmande kärna (eng. pre-emptive kernel) och tilldelning av processor.
- Timers med hög upplösning.
- Låsning av processer i minnet.
- Delat minne.
- Snabba semaforer.
- Kommunikation mellan processer.
- Asynkron I/O.
- Asynkron händelseindikering.
- Kärnprocesser.
- Filsystem med hög prestanda.

D-NIX stöder aktivt ett flertal populära nätverks- och kommunikationsgränssnitt. Detta ger användaren flexibilitet vid kopplingar mot befintliga datorer och kringutrustningar.

036/036/250/Diab Data AB

D-NIX, Unix och Realtid

D-NIX och standard

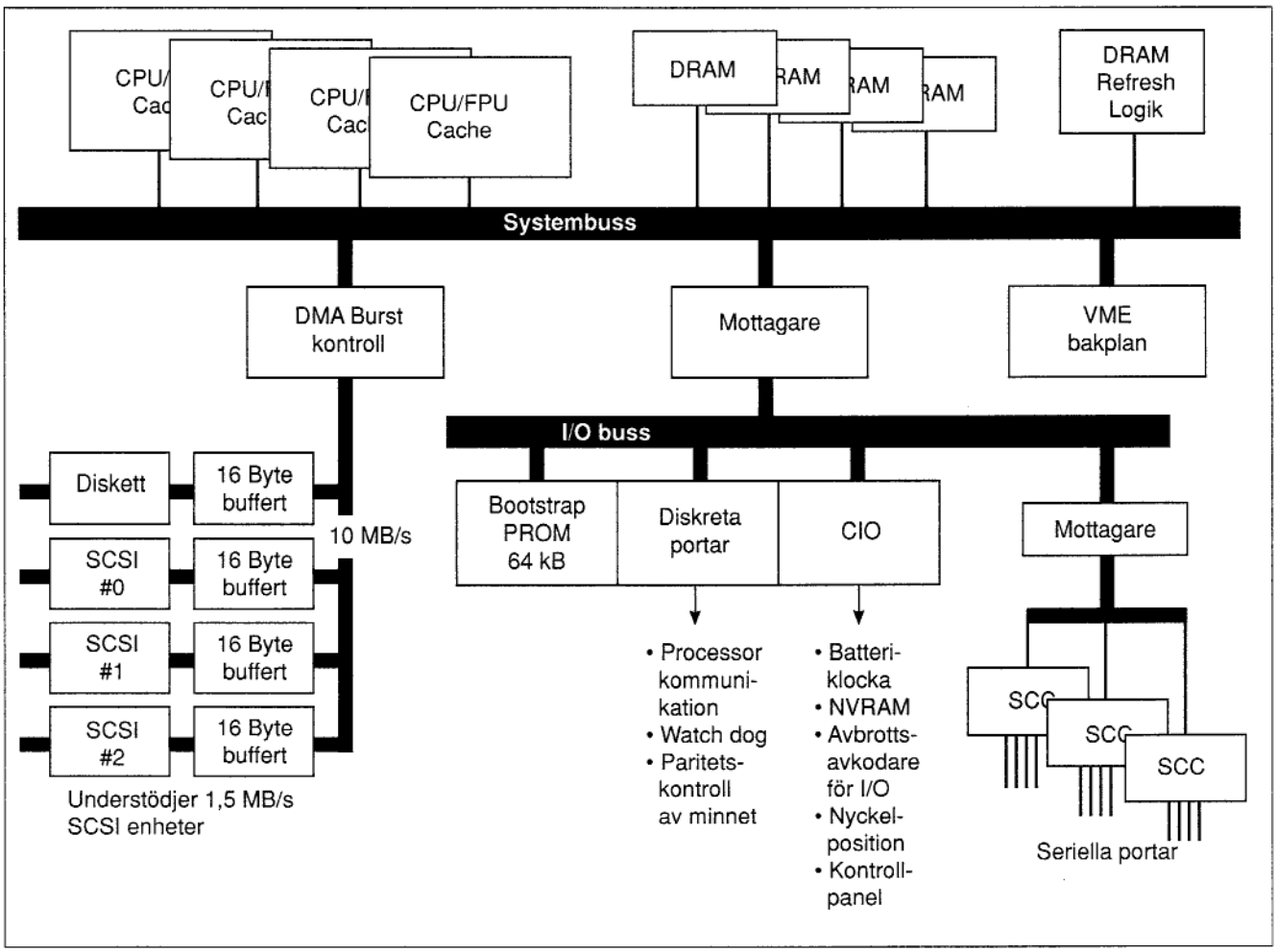
Diab Datas strategi är baserad på att följa standarder både inom maskin- och programvara. D-NIX uppfyller UNIX System V.3 Inteface Definition (SVID) för bassystemet och de utvidgade nivåerna i kärnan samt POSIX 1003.1.

Vårt urval av olika nätverkslösningar baseras på de nätverksstandarder som kommer fram i datorvärlden. DS90 maskin- och programsystem tillhandahåller gränssnitt mot stordatorer från IBM och Unisys liksom gränssnitt mot lokala nätverk med IEEE 802.3 (med fullständig implementering av TCP/IP-protokollet). Det finns även nätverkslösningar för koppling av D-NIX mot PC- eller Macintoshdatorer. Vår maskinvara är baserad på IEEE P1014 standard VME-bus arkitekturen med Motorola 680X0-processorer.

En kombination av maskinvara/programvara

DS90 datorerna är flexibelt uppbyggda och lätta att bygga ut med ytterligare enheter.

Trots att detta är en beskrivning av programvara anser vi att den avancerade maskinvarulösningen har ett intimt förhållande till operativsystemet D-NIX. Maskinvarulösningen i DS90 utnyttjar flerprocessoregenskaperna i D-NIX till fullo. Av den anledningen kommer vi även att diskutera implementeringen av maskinvaran för att ge en mer fullständig bild av dessa system.



Många VME-baserade system är uppbyggda av moduler där varje funktionskomponent finns på ett eget kort. Vår målsättning var att tillhandahålla fullständig funktionalitet på ett kort där utökningar hanteras av moduluppbyggda dotterkort. Som framgår av figuren på föregående sida är möjligheterna på detta enda kort imponerande - upp till fyra stycken MC68040 (var och en med 64 kByte cacheminne), upp till 128 MByte minne, tre snabba SCSI-gränssnitt och fyra seriella gränssnitt.

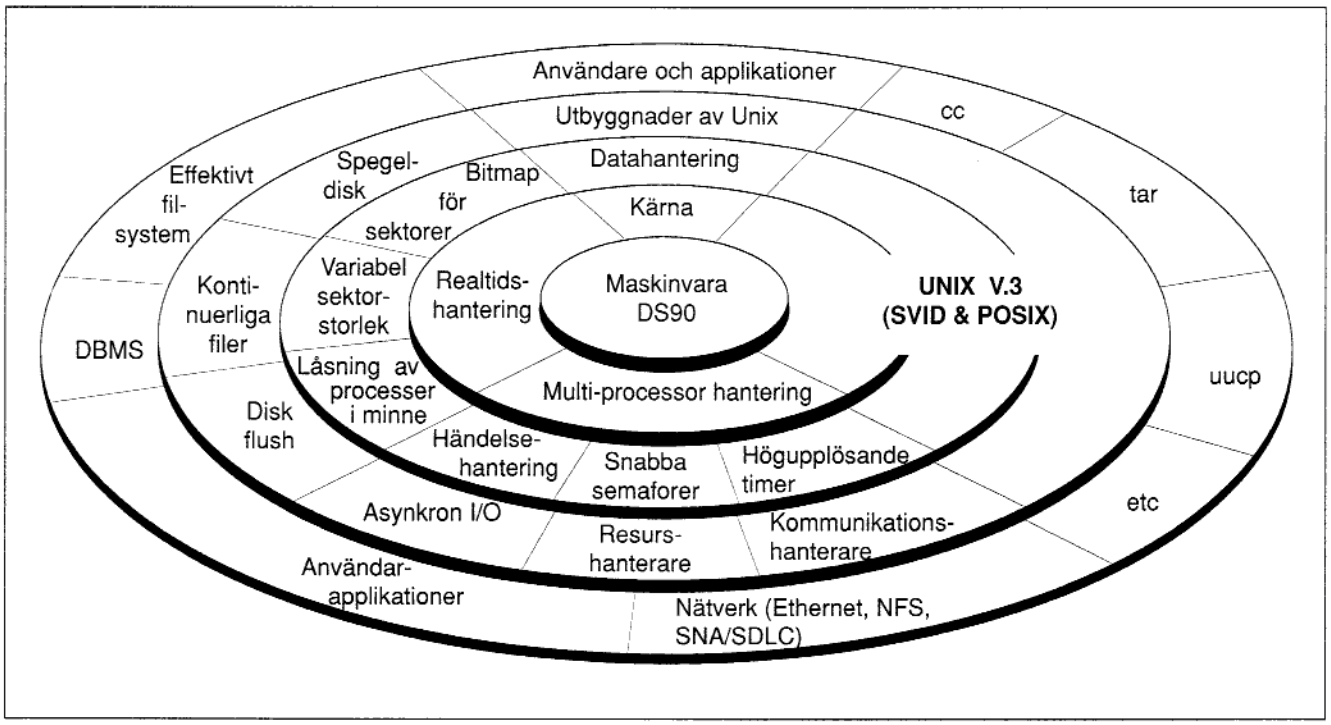
Strukturen i D-NIX

D-NIX eleganta programvarustruktur erbjuder en homogen miljö där UNIX V.3 kombineras med de utökningar i prestanda och realtidshantering som beskrivs i detta dokument.

D-NIX struktur kan beskrivas som olika nivåer av funktionalitet. Den "lägsta" nivån är kärnan där all interaktion med maskinvaran sker tillsamans med programvaran som ansvarar för att dela upp arbetet mellan flera processorer. Den "högsta" nivån är den där applikationerna exekverar. Det är på denna "högsta" nivå som användaren kan skriva program kompatibla med UNIX V.3 SVID men med en betydligt högre prestanda än på de flesta UNIX-system. Med D-NIX kan användaren t ex skriva program som utnyttjar On-Line Transaktionshantering och realtid.

Mellan de yttre nivåerna är D-NIX uppdelat på datahantering och kontrollkod. Det är här som D-NIX får sina unika karaktärsdrag. Det är datahanteringen (nivå 2) och utökningen av UNIX (nivå 3) som ger D-NIX dess förbättringar i prestanda, realtid och On-Line Transaktionshantering.

D-NIX struktur kan beskrivas med fyra nivåer, kärna, datahantering, utbyggnader och applikationer.



Ofta är det så att utökningar runt en standard realiseras på så sätt att effektiviteten av standarden försvagas. Så är inte fallet med D-NIX. Funktionerna i D-NIX, så väl över som under standard UNIX, har implementerats genom att lägga till endast en extra funktionskod till uppsättningen av UNIX V.3 SVID systemanrop. Denna extra funktion är helt enkelt *dnix* där de övriga utökningarna i D-NIX är underfunktioner till *dnix*.

Resultatet av detta är att skillnaden mellan standarden och utökningen är minimal och begränsad. Detta ger både standarden och utökningen obegränsad flexibilitet för ändringar. Exempelvis så erbjuder D-NIX metoder för realtid redan nu. I framtiden kommer liknande metoder för realtid att finnas i standard UNIX. Dessa båda metoder kommer då att kunna samexistera tack vare flexibibliteten i D-NIX.

Vad är realtid ?

Det finns ingen absolut definition för termen realtid. Följande är en definition av realtid som lagts fram av /usr/group tekniska kommite:

Realtid är tillgång till nödvändig prestanda inom en definierad eller begränsad svarstid orsakad av asynkrona yttre händelser.

Vi tänker inte försöka att definiera realtid ytterligare en gång. Istället kommer realtid att behandlas med avseende på UNIX-världen och den lösning som erbjuds via D-NIX.

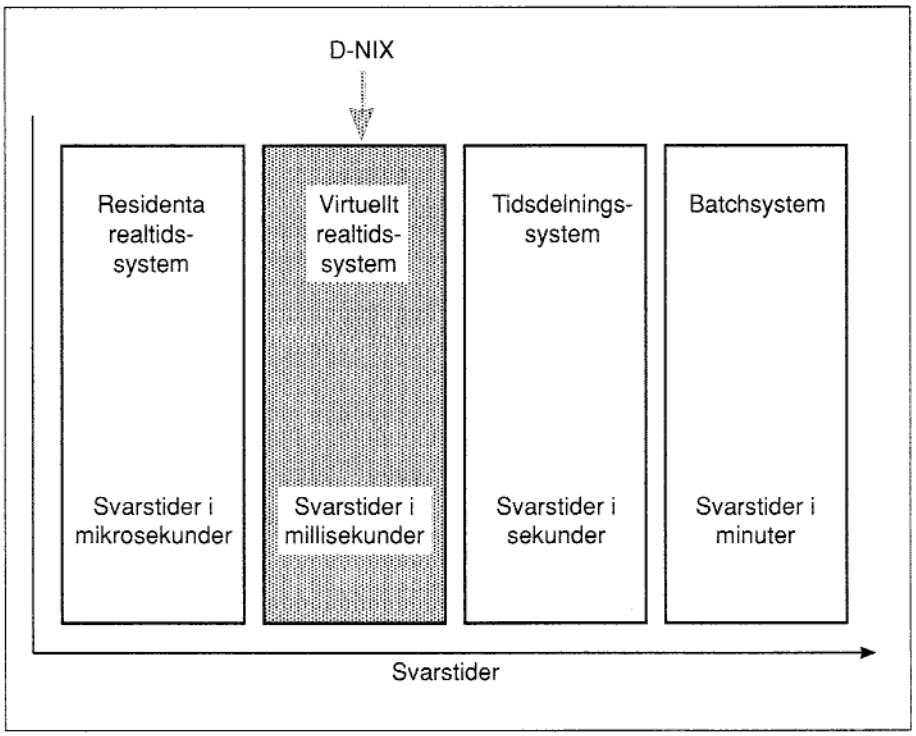
Här kommer vi att definiera och beskriva de frågeställningar som idag möter den som utvecklar realtidstillämpningar. Senare i detta dokument kommer vi att diskutera hur D-NIX realiserar sin målsättning att erbjuda realtidslösningar och samtidigt vara UNIX-kompatibelt. Vidare erbjuder D-NIX en plattform för implementering av realtidslösningar i en ännu mera krävande flerprocessormiljö.

Sedan introduktionen 1984 har operativsystemet D-NIX tillfredställt de olika behoven hos realtidsapplikationer. Fortgående förbättringar utökar dess möjlighet att tillfredställa många realtidsbehov både vad gäller ekonomi och effektivitet.

Översikt

Ett realtidssystem måste kunna svara deterministiskt (inom förutsägbar tid) varje gång ett program- eller maskinvaruavbrott inträffar. Exakt hur mycket "realtid" ett system måste prestera beror på de svarstider som krävs. Riktigt krävande realtidsapplikationer kan begära hela systemets prestanda för att kunna utföra en uppgift "i tid". Detta till skillnad från ett batch-system som endast behöver se till att en uppgift blir utförd "någon gång".

Figuren nedan beskriver fyra nivåer på svarstider representerade i mikrosekunder, millisekunder, sekunder och minuter.



D-NIX har svarstider som ligger nära de residenta realtidssystemen.

036/036/250/Diab Data AB

D-NIX är långsammare än de renodlade realtidssystemen men betydligt snabbare än standard UNIX.

Till en viss del är det sant att "UNIX och realtid är oförenliga". UNIX-standarden kan inte ge deterministiska svarstider vid ökande systembelastning. UNIX utvecklades som ett tidsdelat operativsystem. Det erbjuder många användare tillgång till ett enda system med hjälp av tidsdelning. Det är viktigt i UNIX att alla applikationsprocesser får exekvera "någon gång".

Svarstider och prestanda går hand i hand i en realtidsmiljö. I en tidsdelad miljö kan prestanda vara tillgänglig utan hänsysn till svarstid. Fördelen med ett tidsdelat system är att användarprioriteten inte kontrolleras. Systemet är dynamiskt och erbjuder tjänster till alla användare även om systemet är överbelastat. Ett tidsdelat system i en UNIX-miljö kan vara snabbt om användarna och dess aktiviteter kontrolleras så att systemet aldrig blir överbelastat.

I förbättringarna av D-NIX finns det kontrollmekanismer där specifika processer kan klassas som primära (realtid) och andra klassas som sekundära (tidsdelande). Med denna uppsättning kommer prestanda för de primära processerna att vara konstant även under varierande systembelastning. Resultatet är att endast de sekundära processerna tappar i prestanda och detta endast under höga belastningar av systemet. Detta kommer naturligtvis inte att fungera om hela systemets prestanda skulle användas av primära processer.

Följande faktorer påverkar svarstider:

- Avbrottsfördröjningar (tiden mellan en extern händelse och systemets svar)
- Processprioritet och tilldelning av processor
- Processers förtursshantering
- Processsynkronisering

Följande faktorer påverkar prestanda:

- Rå dataöverföringshastighet
- Filhantering
- Resurshantering
- I/O prestanda

Dessa faktorer är inre komponenter i alla operativsystem. Konsten är att förstå dess samband och interaktion vad gäller realtid. I D-NIX har man löst de begränsningarna som dessa komponenter orsakar, allt inom de ramar som gäller för ett UNIX-kompatibelt operativsystem.

Svarstider

Svarstider är endast giltiga om de är förutbestämbara och kan upprepas. Här kommer vi att jämföra D-NIX med ett rent realtidsoperativsystem.

En mängd faktorer påverkar möjligheterna till korta svarstider i ett realtidssystem. Den mest kritiska faktorn är avbrottsfördröjningar. Detta är den tidsfördröjning som uppstår mellan det att ett avbrott kommer in eller en viss händelse inträffar och det att den programvara som skall hantera avbrottet börjar exekvera. Andra

faktorer (som processprioritet, tilldelning av processor, processers förtur och processsynkronisering) har en direkt påverkan på fördröjningen.

Ett renodlat realtidssystem är utformat för att hålla nere fördröjningstiderna till ett absolut minimum. Detta på bekostnad av generell flexibilitet. I D-NIX hanteras processprioritet och tilldelning av processor via en fast realtidsalgoritm där den process som har högsta prioritet får tillgång till processorn tills den är klar. Tiden för processbyte i processorn minskas genom att man ser till att när en process med högre prioritet finns att exekvera kan den snabbt ta över processorn. I ett renodlat realtidssystem är allting minnesresident vilket gör det enkelt att få snabbhet i processynkroniseringen.

I D-NIX har fördröjningarna minskats avsevärt i förhållande till standard UNIX. Fördröjningstiderna vid processbyte i processorn är upp till 1000 gånger kortare än hos de flesta UNIX-system. Processprioriteten och tilldelning av processor finns i två uppsättningar - en för UNIX tidsdelande processer och en för realtidsprocesser. Tiden för att byta process i processorn är deterministisk eftersom D-NIX opprativsystem är uppdelat i sektioner och den längsta tiden för processbyte är lika med tiden för den längsta sektionen. Nya funktioner har kommit till för att tillhandahålla snabba minnesmekanismer som gör det möjligt för olika processer att synkronisera med varandra. Resultatet av detta är svarstider som är många gånger snabbare än i standard UNIX-system.

Prestanda

Realtidsprestanda är något som det sällan finns tillräckligt mycket av för att tillfredställa användarnas behov. Mikroprocessorarkitektur och förbättringar vad gäller snabbhet är de största anledningarna till att systemen blir snabbare. Den största flaskhalsen vad gäller prestanda ligger i I/O - speciellt mot massminnen. Här har inte maskinvaran kunnat uppfylla de ökade prestandakraven från användarna till ett rimligt pris.

I D-NIX har det lagts ned mycket arbete på att förbättra I/O-prestanda men fortfarande behålla UNIX-kompatibiliteten. För användaren ser filsystemet i D-NIX ut på samma sätt som i standard UNIX, men vad gäller prestanda är det betydligt bättre. D-NIX tillhandahåller även en speciell funktion kallad hanterare (eng. handlers) vilka ger ökad snabbhet och flexibilitet vid olika typer av I/O - något som inte finns under UNIX.

Flexibla lösningar

D-NIX ger en flexibel lösning på behoven av realtidshantering. Lösningen i D-NIX har inte den snabbhet som finns i rena realtidssystem men den erbjuder betydligt kortare svarstider än Unix tillsammans med UNIX-kompatibilitet och flexibilitet som inte fanns tidigare.

Processprioritet och tilldelning av processor finns för både tidsdelande processer och realtidsprocesser. Stöd för flera processorer med fördelad belastning, ett filsystem med hög prestanda samt UNIX-kompatibilitet är grundstenar i D-NIX. Dessutom har D-NIX resurshanterare (som för tillfället inte finns i UNIX).

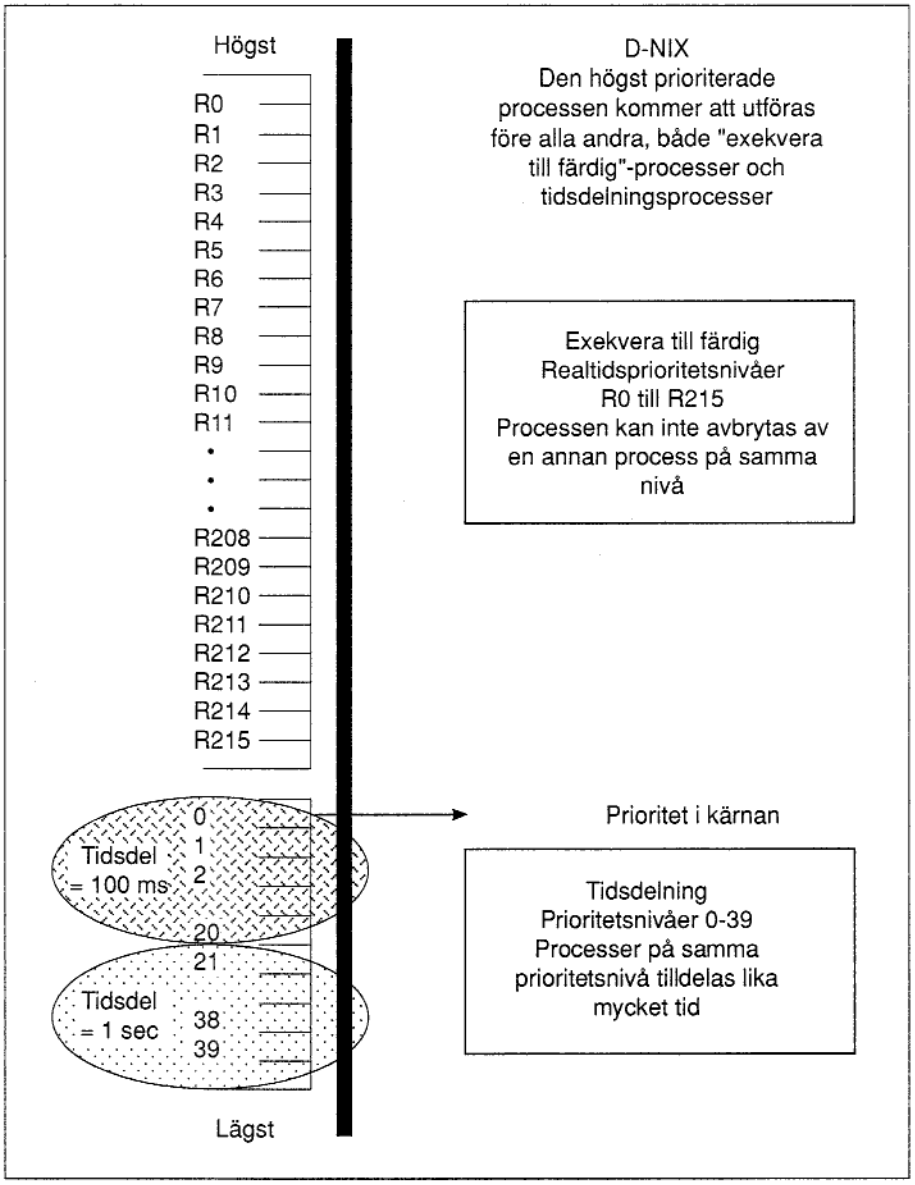
036/036/250/Diab Data AB

D-NIX Högprestandatillägg

Processprioritet och tilldelning av processor

Så fort en process är klar att exekvera är det dess prioritet som avgör när detta sker. Under UNIX-baserade, flerprocesssystem, exekverar varje processor endast en process åt gången. Alla andra processer som är klara att exekvera läggs i en "färdig att exekvera"-kö. Denna kö sorteras efter prioritet med den process som har högst prioritet först.

Förutom de vanliga prioriteterna som finns i UNIX har D-NIX 215 realtidsprioritetsnivåer.



Som framgår av figuren kan processfördelning specificeras att utföras på två sätt - UNIX-kompatibel tidsdelning eller D-NIX realtid "exekvera tills färdig". D-NIX kombinerar dessa vilket totalt ger 256 prioritetsnivåer. Där 0 till 39 används för standard UNIX tidsdelning och R0 till R215 används för D-NIX realtid.

Metoden för tilldelning av processorn vid tidsdelning går ut på att varje process på samma prioritetsnivå får tillgång till lika mycket processortid för exekvering. Med denna metod varierar svarstiderna med belastningen på systemet.

Det är tillägget med att kunna "exekvera tills färdig" som ger D-NIX dess realtidsmöjligheter. Detta tillägg gör det möjligt för användare att tilldela prioriteter till processer, vilket i sin tur resulterar i att dessa processer exekverar med svarstider som krävs av realtidsapplikationer. Metoden för tilldelning av processorn för "exekvera tills färdig" ger användaren snabba och bestämbara svarstider. Detta betyder att när en process exekverar kommer den att fortsätta att göra det tills:

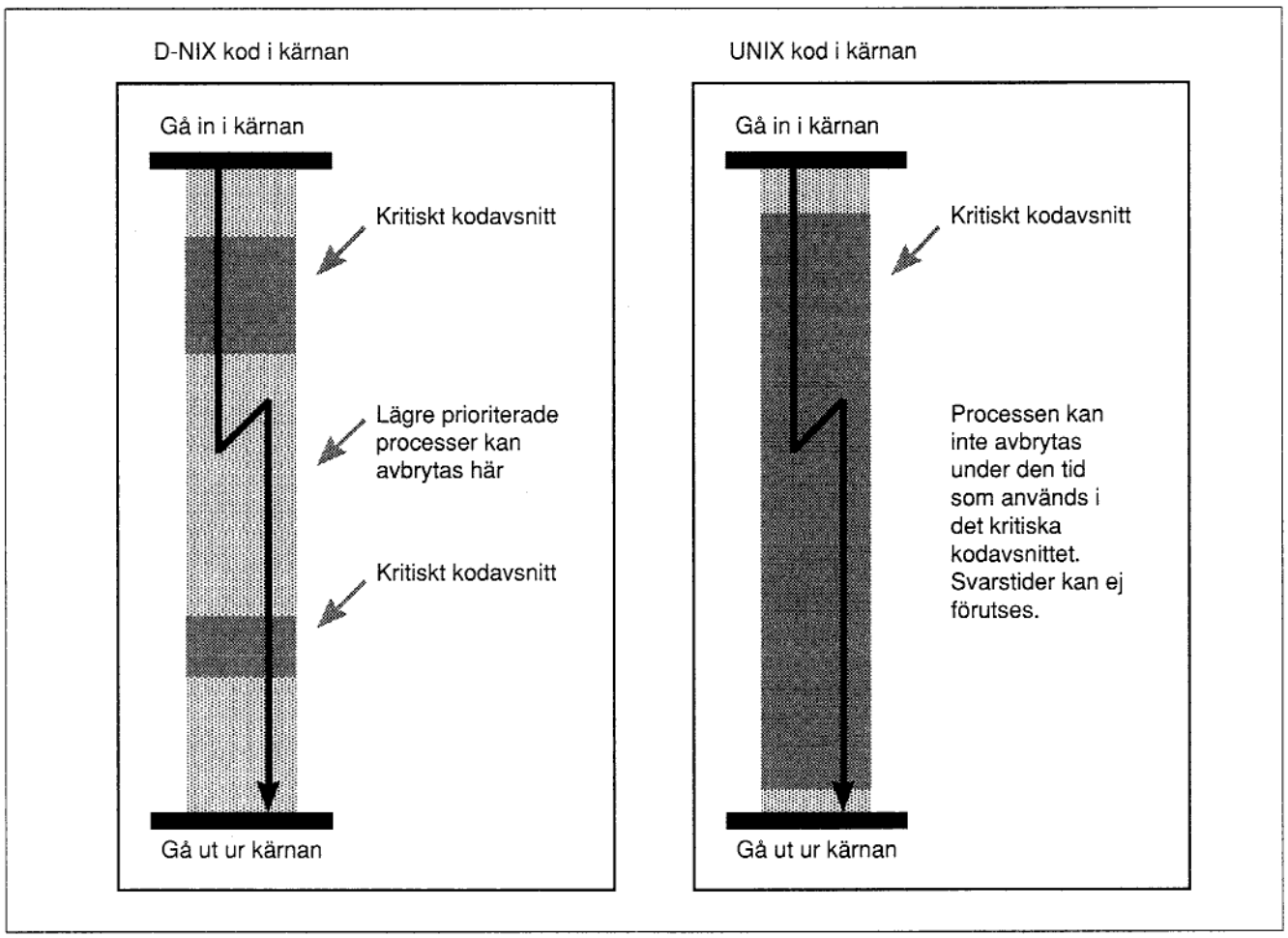
- Den avbryts av en process med högre prioritet.
- Blockeras i väntan på en systemresurs.
- Lämnar tillbaka processorn frivilligt.

En process som för "exekvera tills färdig" har högre prioritet än kärnan i D-NIX. Så fort det finns en process färdig att exekvera med högre prioritet än den som för tillfället exekverar kommer processen med högre prioritet att avbryta exekverande process oavsett tilldelningsmetod för exekverande process.

Kärnan och tilldelning av processor

I kärnan för alla operativsystem finns det kodavsnitt som är kritiska. Dessa kritiska avsnitt innehåller kod där systemdatastrukturer uppdateras. För att upprätthålla integriteten i dessa datastrukturer måste alla operationer på dessa ske utan avbrott. D v s om den exekverande processen befinner sig i ett kritiskt avsnitt får den inte bli avbruten, även om den andra processen har högre prioritet, innan den är klar i det kritiska avsnittet.

I D-NIX är de kritiska kodavsnitten färre och kortare än i UNIX, vilket ger bestämbara svarstider.



036/036/250/Diab Data AB

För att ge D-NIX dess möjligheter med förtur till processorn måste antalet kritiska kodavsnitt (och dess storlek) begränsas till ett minimum. Omvänt kan man se standard UNIX som ett enda kritiskt kodavsnitt vilket gör förtur för andra processer till processorn omöjlig. Detta tillägg till D-NIX försäkrar att så lite tid som möjligt spenderas i kritiska kodavsnitt för att hålla nere fördröjningarna vid förtur till processorn (tiden mellan ett avbrott och det att motsvarande servicerutin startar) till ett minimum. För mer information om kritiska avsnitt och flera processorer, se avsnittet om *Flerprocessorlösningar*.

Låsning av processers minne

Liksom UNIX är D-NIX ett system med virtuellt minne där processer kan läggas ut på skivmine för att frigöra minne åt den exekverande processen.

I ett system med virtuellt minne är det möjligt att en process kod-, data- eller stacksida inte finns i minnet då den behövs - det kan vara så att den aldrig lästs in i minnet eller har laggts ut på skivminne då en annan process har behövt mer minne. Den tid det tar att läsa tillbaka denna minnessida från skivminnet kan vara betydligt längre än vad en händelse kan vänta på svar. Genom att låsa en process i minnet kan D-NIX undvika dessa skivminnesfördröjningar.

En begränsning med standard UNIX är dess timers med dålig upplösning (en sekund). Realtidsapplikationer kräver betydligt bättre funktionalitet och upplösning på timers än vad standard UNIX kan erbjuda. D-NIX erbjuder timers med en upplösning som endast begränsas av upplösningen hos systemklockan. Upplösningen på timers i UNIX är en sekund, men i D-NIX är upplösningen 16,5 millisekunder. Avsnittet om *Avancerade I/O-tjänster* beskriver hur dessa timers kan användas asynkront.

Snabba semaforer

Snabba semaforer kan användas i delat minne för synkronisering mellan processer. D-NIX har tillgång till väldigt snabba binära och räknande semaforer för effektiv synkronisering av processer - till skillnad från semaforerna i standard UNIX. Semaforerna i D-NIX är i genomsnitt två gånger snabbare än i standard UNIX. Avsnittet om *Avancerade I/O-tjänster* beskriver hur snabba semaforer kan användas asynkront.

Semaforkommandon i D-NIX

SETEVENT	sätt värde för en händelsesemafor
WAITEVENT	vänta tills händelsesemafor har angivet värde
SETVAL	sätt värde på räknande semafor
INCVAL	räkna upp värde på räknande semafor
DECVAL	räkna ned värde på räknande semafor

Drivrutiner

D-NIX har en speciell funktion som tillåter en priviligierad användarprocess direkt, exklusiv eller delad tillgång till I/O-minne. Detta ger utvecklare möjligheter för användarprocesser som liknar drivrutiner. D-NIX tillhandahåller ett "C"-bibliotek med gränssnittsrutiner som kan användas av en priviligierad process för åtkomst till externa enheter. D-NIX tillhandahåller även utvecklaren med de normala UNIX-möjligheterna med drivrutiner.

036/036/250/Diab Data AB

Flerprocessorlösning

System med flera processorer mångfaldigar ett operativsystems möjligheter. När systemresurserna är tillräckliga för en viss blandning av applikationer men processorkapacitet saknas är den vanligaste lösningen ett flerprocessorsystem.

Arkitekturen i D-NIX stöder ett stort antal processorer. För tillfället stöder maskinvaran upp till fyra 68040 processorer i ett system. Dessa processorer delar på en egen intern buss med samma tillgång till systemets resurser och åtkomst av VME-bussen. Varje processor har sitt eget 64 kbyte cache-minne.

Pris/prestanda

Ofta kräver en systeminstallation mer processorkraft än vad ett enprocessorsystem kan erbjuda. Fortfarande behövs dock flexibiliteten hos en vanlig processor och operativsystemet. Ett flerprocessorsystem kan då erbjuda den prestanda och funktionalitet som användaren letar efter.

I ett system med två processorer kommer prestandan att nästan fördubblas (beroende på blandningen av applikationer) samtidigt som systemkostnaden endast ökar med kostnaden för en extra processor. Om systemets baskonfigurering är den samma kan man med en extra processor öka systemets prestanda och skriva av den ökande kostnaden mot en betydande ökning i datorprestanda.

För många operativsystem ökar kostnaderna för programvaruutveckling avsevärt då gamla applikationer skall skrivas om för att fungera i en flerprocessormiljö. I dessa fall krävs stora ändringar inte bara för att ge processorerna delad åtkomst till systemdata och resurser utan även för att förhindra att data förstörs vid samtidig uppdatering.

Användarprogram under D-NIX fungerar utmärkt såväl i ensom flerprocessorsystem. Förflyttningen till flerprocessorsystem (oavsett antalet ytterligare processorer) sker utan att användaren lägger märke till det.

Implementeringar av flera processorer

Löst kopplade system jämfört med hårt kopplade system

Med ett löst kopplat system menas ett system med en grupp av processorer där varje processor har en egen uppsättning resurser. Ofta i realtidstillämpningar krävs att en processor är kopplad till en resurs för att kunna erbjuda tillräckligt korta svarstider. Med denna lösning erhålls de kortaste svarstiderna och den största genomströmningen av data.

Om din applikation kräver en extra processor med sina egna resurser är en löst kopplad flerprocessorimplementering den bästa lösningen. En typ av löst kopplad implementering är då separata system samexisterar i ett lokalt nätverk.

Generellt sett avser en hårt kopplad implementering ett enda system där flera processorer delar på en uppsättning operativsystemresurser. En sådan hårt kopplad miljö delar vanligtvis fullständiga operativsystemsfunktioner växelvis. D-NIX är en hårt kopplad implementering.

Diab Datas lösning

Diab Datas flerprocessorsystem använder en hårt kopplad flerprocessorarkitektur. Upp till fyra processorer kan använda systemets samtliga resurser. Varje processor har samma tillgång till minne och andra resurser. I ett sådant hårt kopplat system kan de enskilda processorerna byta av varandra. På så sätt kan en process (eller någon annan del av D-NIX) exekveras av den första processorn som blir ledig. Resultatet av detta blir snabbare svar och ett bättre utnyttjande av av resurserna - en process behöver inte flyttas mellan processorer.

D-NIX operativsystem exekverar på alla processorer. Tidsfördelaren, filsystemet och styrfunktioner exekverar på alla processorer. På så sätt delar processorerna automatiskt på belastningen i systemet.

Under D-NIX utnyttjar användarprogrammen flerprocessormiljön via den normala processtrukturen. Att systemet använder flera processorer påverkar inga kommandon, bibliotek eller kompilatorer eftersom D-NIX normalt ger användarnivån uppfattningen att flera processer exekverar parallellt. Det är endast kärnan i D-NIX som känner till hur många processorer det finns i systemet. Hanteringen av flera processorer är helt transparent för användaren. Programvaran är helt flyttbar mellan alla DS90 en- eller flerprocessorsystem.

Följande är en beskrivning av utökningarna i D-NIX jämfört med standard UNIX. Dessa var nödvändiga för att kunna hålla de parallella flerprocessorfunktionerna transparenta för användaren:

- Process/processor och prioritet
- Överensstämmelse cache/systemminne
- Processorsynkronisering
- Avbrottshantering

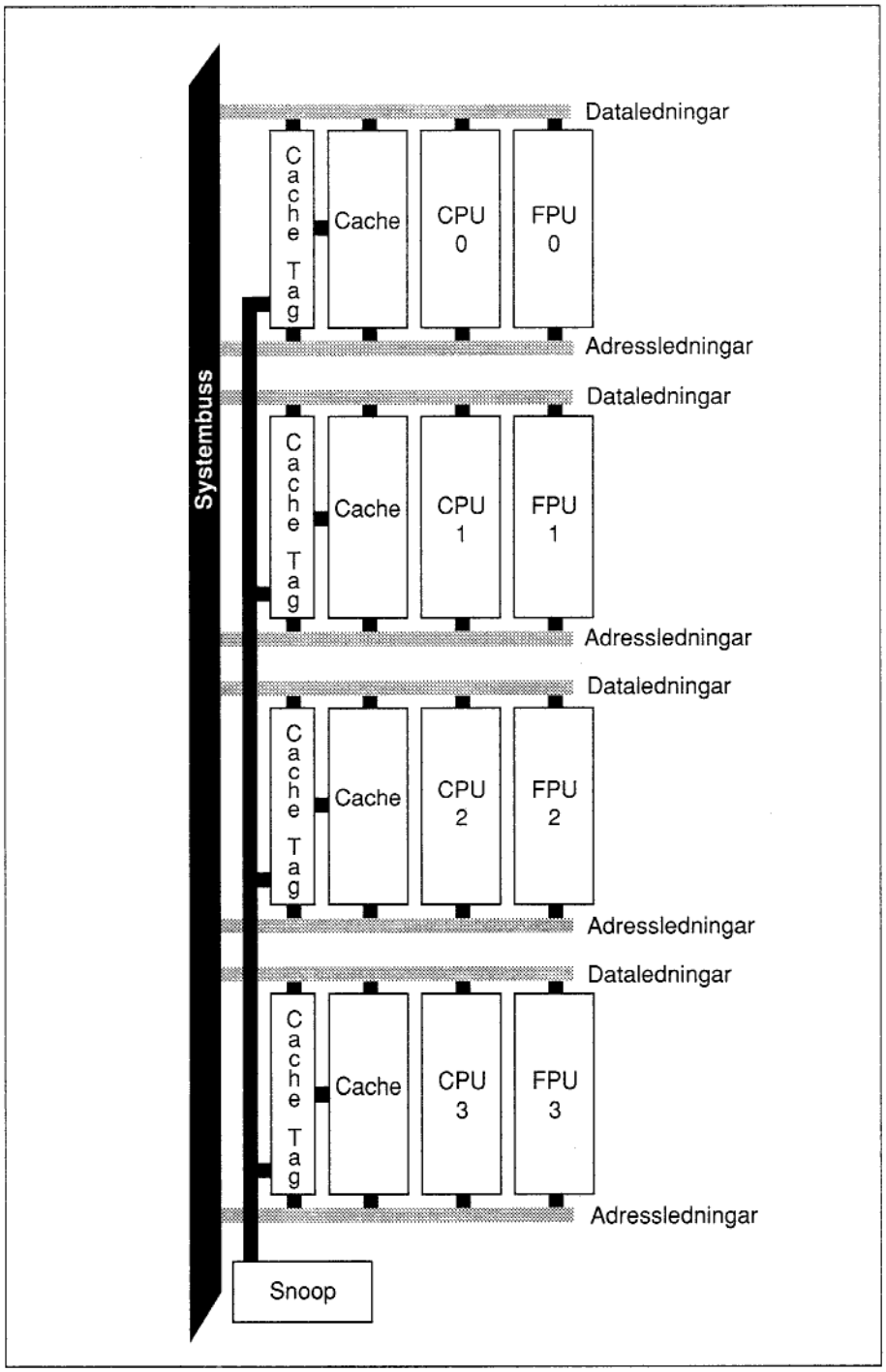
Processprioritet: I realtids- och högprestandamiljöer är processprioriteten av högsta betydelse. Den får ytterligare en dimension, processornivå, i ett system med flera processorer. När det kommer in en avbrottssignal (som skall behandlas av en servicerutin) letar tilldelaren av processorer efter en ledig processor. Om alla processorer är upptagna kontrollerar tilldelaren av processorer om någon av de exekverande processerna har en prioritet som är lägre än prioriteten för den väntande servicerutinen.

I så fall kommer den exekverande process som har lägst prioriteten att bytas ut mot den väntande servicerutinen. Om ingen exekverande process har en lägre prioritet kommer den väntande processen att placeras i D-NIX "färdig att exekvera"-kö i prioritetsordning.

Överensstämmelse cache/systemminne: Allt eftersom mikroprocessorer har blivit snabbare har en mekanism kallad "caching" utvecklats för att öka hastigheten på processorns åtkomst av minne. Detta i syfte att förbättra systemens prestanda.

Ett cache-minne är ett snabbt minne intimt kopplat till varje processor i DS90-datorerna. Varje 68040 har sitt eget 64 kbyte cacheminne. Under D-NIX hanteras varje 64 kbyte cache som 4096 cacherader. Varje cache-rad består av fyra långa ord (16 byte) samt en

Cach-minnet ger systemet betydligt bättre prestanda.



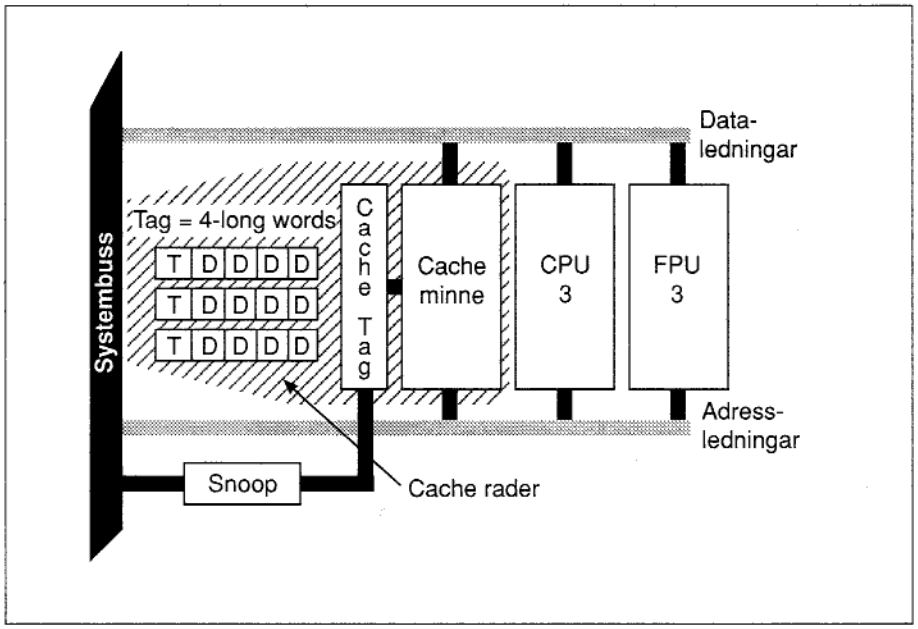
etikett som indikerar om och vart ifrån i minnet cache-raden har kopierats (se figur ovan och på nästa sida).

Cache-minnet gör att man får tillgång till 16 byte nästan lika snabbt som en enda byte. Att ladda cache-raderna med 16 byte på en gång reducerar antalet referenserna till det delade systemminnet. Det ger även möjlighet till att "kika i förväg" så att när processorn behöver minne finns det redan tilgängligt i cache-minnet.

När processorn behöver tillgång till en viss adress i minnet kontrollerar den först om adressen redan finns i cache-minnet. Om den

gör det är det en cache-"träff" och exekveringen fortsätter. Om den inte gör det kommer innehållet i minnesadressen att läsas in till cache-minnet (cache-"miss"). En flagga, gällande/icke gällande, sätts eftersom minnet är globalt och kan modifieras av vem som helst - om en modifierad minnesadress även finns i ett cache-minne skall cache-raden markeras som icke gällande. Ett vanligt exempel på detta är då delat minne används i UNIX och mer än en processor modifierar samma minnesblock. Förflyttningar av data från systemminnet till cache-minnet utförs i sk "block burst mode". VME I/O-minne flyttas inte till cache-minnen.

I DS90 datorarkitektur möjliggörs överensstämmelse mellan cache- och systemminne via två hårdvarufunktioner - "Write-Through Cache" och "Cache Snoop".



Med Write-Through Cache och Cache Snoop garanteras integriteten i minnet.

Write-Through Cache: Varje gång en skrivning sker till en adress som finns i cache-minnet uppdateras även systemminnet. Om denna adress även finns i en annan processors cache-minne flaggas denna cache-rad som icke gällande.

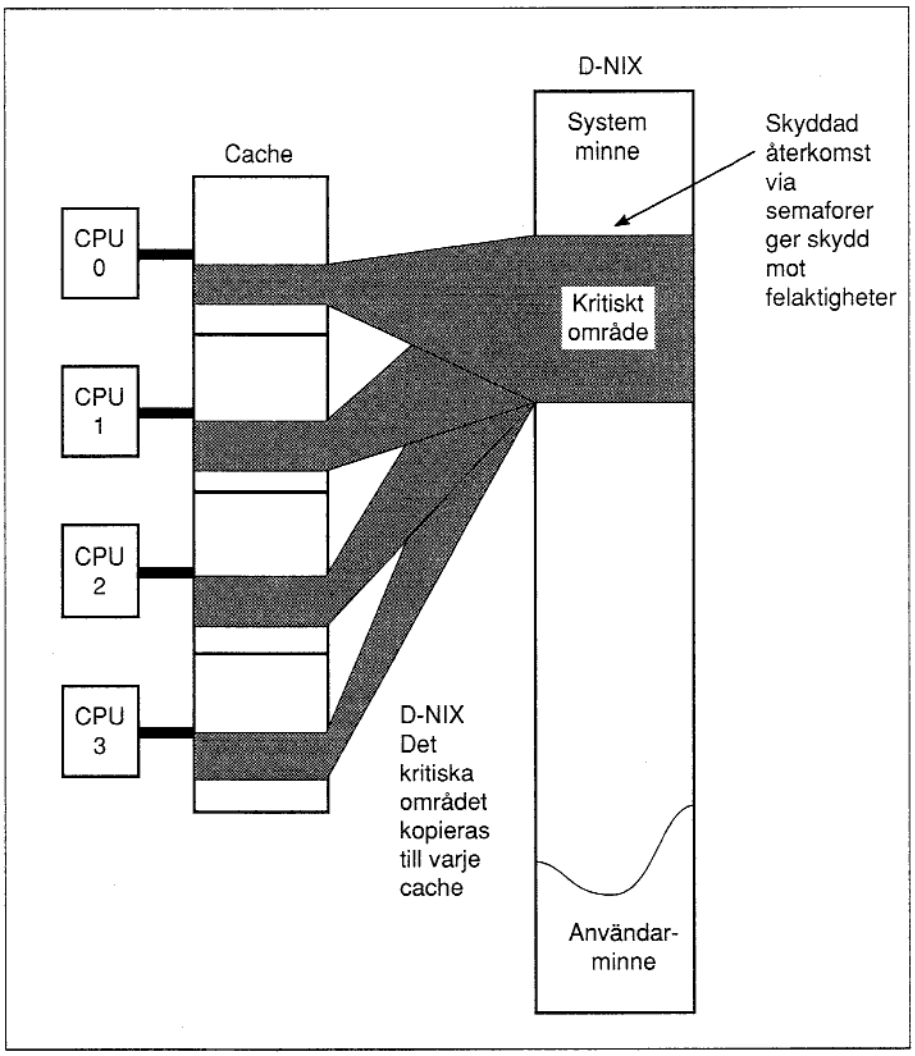
Cache Snoop: "Snoop"-kretsen lyssnar på alla skrivningar till systemminnet. Skrivningarna kan komma vart som helst ifrån - vilken processor som helst, via DMA eller VME-bussen. När en adress som finns i något cache-minne uppdateras flaggar "Snoop"-kretsen aktuell cache-rad som icke gällande.

Processorsynkronisering: I D-NIX exekverar flera processer (en per procesor) helt asynkront till varandra vid varje tidpunkt. Dessa processer måste ha möjlighet att kunna utesluta varandra. Detta är nödvändigt för att vissa kritiska resurser skall skyddas mot oavsiktlig modifiering på grund av samtidig åtkomst (se avsnittet om *D-NIX Högprestandatillägg*).

D-NIX isolerar kod som har tillgång till kritiska resurser i s k kritiska regioner. Kärnan är den enda del av operativsystemet D-NIX som använder kritiska resurser. Ett exempel på en kritisk resurs är en kö (t.ex "färdig att exekvera"-kön) som används av processortilldelaren för att avgöra vilken process som skall exekvera nästa gång. När en processor exekverar koden för kärnans

036/036/250/Diab Data AB

Semaforer används för att skydda kritiska regioner mot felaktigheter.



tilldelare (och arbetar mot "färdig att exekvera"-kön) och en annan processor vill göra samma sak kommer D-NIX att förhindra den andra processorn att göra detta tills den första processorn är klar med användningen av "färdig att exekvera"-kön.

Semaforer tillhandahåller programvaruskydd av kritiska resurser. En semafor fungerar som en nyckel/lås vilken ger sekventiell åtkomst för att skydda en kritisk region. Varje kritisk resurs tilldelas en egen unik semafor som garanterar att endast en processsor får tillgång till semaforen.

I föregående kapitel beskrev vi kritiska regioner. I D-NIX hanteras kritiska resurser inom kritiska regioner. I teorin kan dock en kritisk region existera utan att det finns en åtföljande kritisk resurs. Åtkomst och kontroll av både kritiska regioner och resurser hanteras av unika semaforer.

Avbrottshantering: I D-NIX kan drivrutiner exekveras av samtliga processorer. Undantaget är drivrutinens avbrottsrutin - den exekveras endast av den första processorn. Den första processorn i systemet kan skicka avbrottssignaler till övriga processorer.

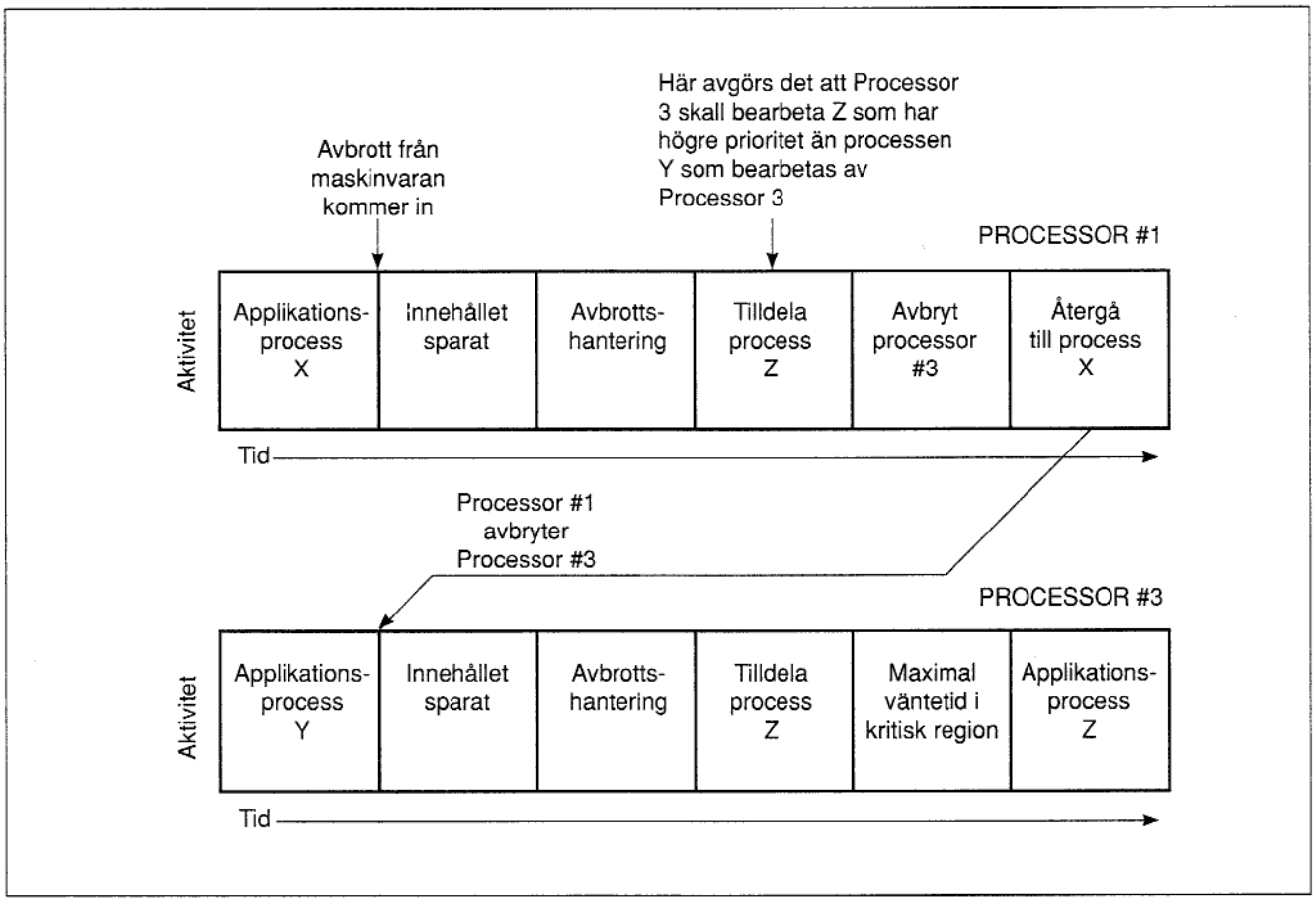
För t ex en terminalhanterare kan vilken processor som helst begära att få skicka en textsträng till terminalen. Men om operatören trycker på en tangent kommer maskinvaran för terminalens gränssnitt att skicka en avbrottssignal till processorn för att indikera att ett tecken kommit in. Denna avbrottshantering kommer att skötas av den första processorn.

Efterföljande tilldelning av CPU för den process som väntar på det inkommande tecknet från terminalen kommer också att hanteras av den första processorn.

Om vi fortsätter med exemplet i figuren nedan (för att avsluta hanteringen av det inkommande tecknet) så exekverar processor nummer tre en process med lägre prioritet och avbryts av den första processorn. Processor nummer tre går igenom tilldelaren och får reda på att dess aktuella process skall fråntas processorn och att den skall hämta in processen som skall hantera det inkommande tecknet.

Den tid det tar för en process att börja exekvera efter en avbrottssignal är ett mycket kritiskt värde i ett realtidssystem. Den genomsnittliga fördröjningstiden (avbrott till startad process) för aktuell hårdvara (68040) och D-NIX 5.3 är mindre än 5 millisekunder.

Processor 1 avgör om en process på en annan processor skall avbrytas och exekvera en annan process.



036/036/250/Diab Data AB

Sofistikerad filhantering

För att göra D-NIX till ett UNIX-baserat system med hög prestanda och dataintegritet har förbättringar gjorts i filhanteringen. Ett snabbt flerprocessorsystem tappar prestanda om det bromsas upp vid arbete mot filer. Detta gäller speciellt om det är ett system för realtidsapplikationer. D-NIX löser detta genom förbättringar i maskin- och programvaran som kompletterar varandra:

På maskinvarusidan:

- Flera SCSI-gränssnitt
- Höghastighets DMA-gränssnitt (disk till minne)
- SMD-gränssnitt som tillval

På programvarusidan:

- Kontrollerade diskskrivningar
- Bit-mapp över använda disksektorer
- Kontinuerlig diskallokering för filer
- Variabel sektorstorlek
- Spegeldiskar

Disk I/O

Maskinvaran som D-NIX exekverar på har fått följande förbättringar för att erbjuda optimal disk I/O prestanda:

Flera SCSI-gränssnitt: Tre SCSI-gränssnitt är intimt kopplade till processor(erna) och minnet. Med ett enkelt SCSI-system kan det finnas stöd för ett flertal olika enheter men endast ett kan användas åt gången vilket begränsar användningen av system I/O. För DS90 systemen är det inte så, där kan upp till tre separata I/O-enheter samtidigt utföra I/O.

Tack vare denna förbättring kan D-NIX flytta data från disk till band med full hastighet. Både disken och bandstationen har separata SCSI-kontrollenheter och "pipen" för databufferten innehåller alltid data tack vare D-NIX asynkrona I/O-möjligheter. Asynkron I/O diskuteras senare. Ett speciellt tillval har laggts till "tar"-kommandot för att specificera denna fullhastighetsöverföring.

Gränssnitt för höghastighetsöverföring disk till minne: D-NIX arbetar på ett VME-baserat system vilket gör att SCSI-kontrollenheter kan användas med tillgång till minne via VME. Som framgår av tidigare figur använder D-NIX maskinvara där processorerna, minnet och SCSI-kontrollenheterna är intimt kopplade. I/O-bussen (10 MB/sek) mellan diskgränssnitten och minnet möjliggör snabb DMA-åtkomst och eliminerar overhead i jämförelse med andra mer traditionella VME-baserade SCSI-kontrollenheter.

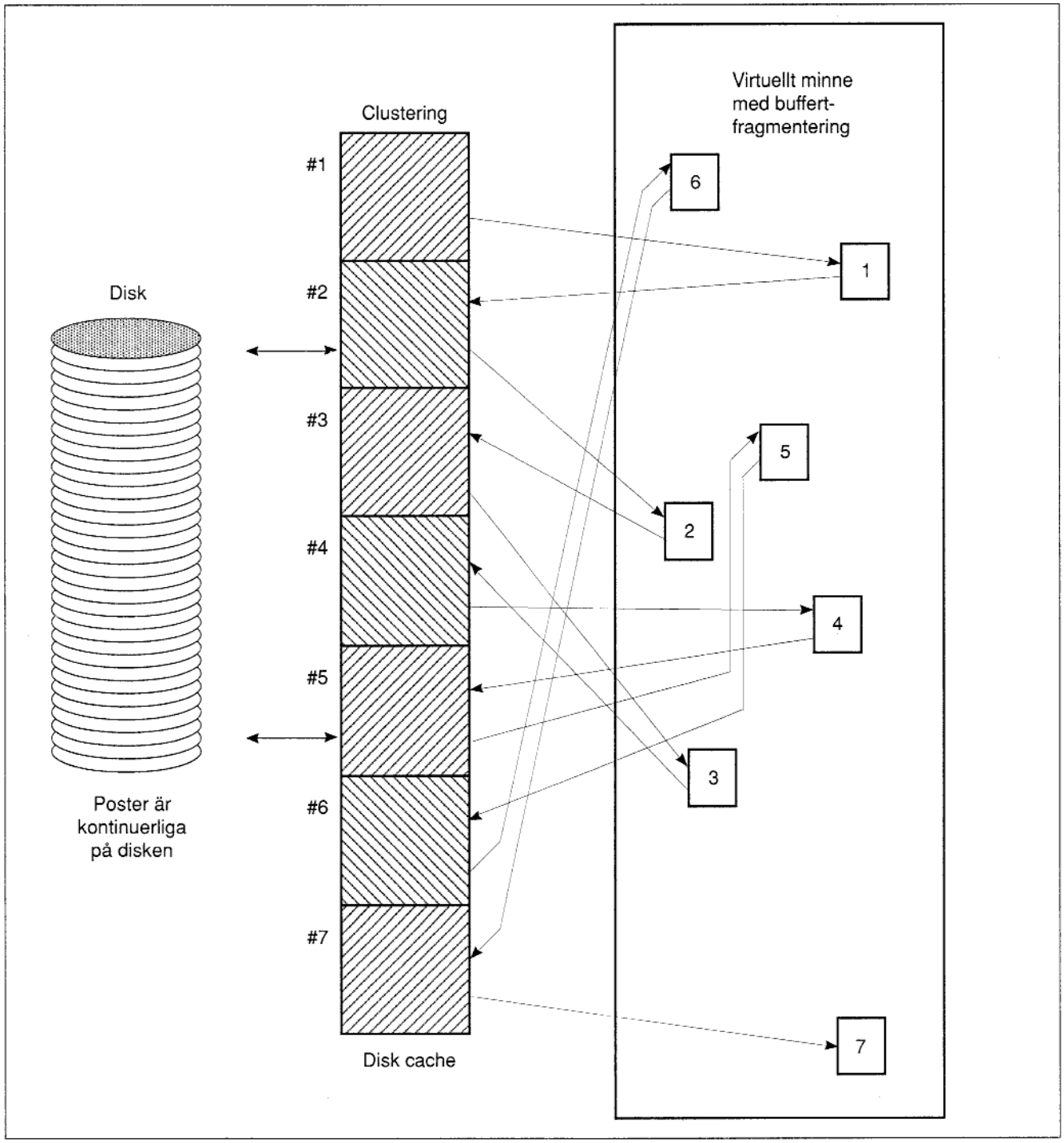
SMD-gränssnitt som tillval: För att kunna stödja större diskkapacitet per volym (> 1 GB) stöder D-NIX SMD-diskkontrollenheter. SMD-kontrollenheten placeras på VME-bussen vilket gör att den kan användas utöver de tre befintliga SCSI-kontrollenheterna.

Förbättrat filhanteringssystem

Förutsägbara och snabba åtkomsttider på disken är ett grundkrav i realtidssystem med hög prestanda. Det är viktigt att kunna förutsäga hur lång tid det tar att få tillgång till data på disken och hur snabbt data överförs mellan disk och minne. Följande används för att uppfylla dessa krav:

Kontrollerade diskskrivningar: I UNIX, när en applikationsprocess utför en skrivoperation mot disken, är det operativsystemet som övertar ansvaret för att data verkligen skrivs till disken. Problemet i UNIX är att kontrollen återgår till applikationen (efter skrivoperationen) vilket gör att applikationen aldrig vet om den fysiska skrivningen verkligen har gjorts till disken. I D-NIX (för att komma förbi detta) tvingas alla databuffertar för en specifik fil att

Utspridda minnesbuffertar kan "kedjas" samman för snabb överföring till disken.



036/036/250/Diab Data AB

skrivas ut på disken vid systemanrop för skrivning och applikationen återfår inte kontrollen förrens skrivningen är avslutad eller ett fel uppstått.

Bit-mapp över använda disksektorer: Standard UNIX använder en länkad lista för lediga disksektorer. Denna länkade lista över lediga disksektorer tenderar att sprida data över disken efter en tid. Detta leder till ökade åtkomsttider då den länkade listan innehåller sektorer spridda över disken.

D-NIX använder en bit-mapp för använda disksektorer. En bit-mapp fungerar som en sorterad lista över lediga sektorer och förhindrar ökade åtkomsttider till disken. Man behåller dock full kompatibilitet med UNIX filsystem.

Kontinuerlig diskallokering för filer: Diskoperationer i standard Unix betraktas som slumpmässiga åtkomstoperationer. Detta börjar vid skapandet av en fil. Inledningsvis allokeras en mindre del diskutrymme. Allteftersom filen växer kommer ytterligare utrymme att allokeras någonstans på disken. När detta har upprepats ett antal gånger kommer åtkomsttiderna att vara oförutsägbara - tid går åt till disksökningar över en stor del av disken för att komma åt sekventiella applikationsdata.

I D-NIX finns ett alternativ till detta - kontinuerlig diskallokering för filer. Genom att allokera utrymme för en fil i förväg, med en storlek anpassad till applikationen, kan åtkomsttiderna bli förutsägbara och den överförda datamängden så stor som möjligt. Om gränsen för en förallokerad fil överskrids kommer filens storlek automatiskt att justeras av D-NIX.

Clustering: I ett system med virtuellt minne, som UNIX, är minnesbuffertar som används för disk I/O normalt utspridda i minnet. En normal dataöverföring från eller till disk går till så att diskkontrollenheten får en adress i minnet tillsammans med antalet bytes som skall överföras. Resultatet av detta är att diskkontrollenheten avbryter systemet efter varje överförd databuffert (det kan krävas hundratals överföringar disk/buffer för en disk I/O-operation). Detta är mycket ineffektivt. För det första blir systemet avbrutet många gånger och för det andra, då systemets processor ger diskkontrollenheten en ny adress och antal bytes som skall överföras, fortsätter disken att snurra (vilket kräver ett fullt varv för nästa läsning/skrivning). Detta reducerar dataöverföringen till disk avsevärt.

Maskinvaran för DMA på Diab Data's processorkort tillåter att buffertar "kedjas" samman. Detta gör det möjligt att gruppera de utspridda minnesbuffertarna för överföring i en diskoperation. D-NIX diskhanterare försöker se framåt om det finns block på disken som ligger intill varandra. Om de gör det kommer minnesbuffertarna att kedjas samman för att möjliggöra en oavbruten DMA-överföring - vilket ger en snabb 1:1 "disk interleaving"-faktor. Om diskblocken inte ligger intill varandra kommer "kedjningen" inte att vara genomförbar vilket leder till dålig diskprestanda.

Variabel blockstorlek: Den ideala blockstorleken i ett filsystem för optimal prestanda varierar från applikation till applikation. Exempelvis skulle en databas med en poststorlek på 8 kByte förmodligen exekvera snabbare med en blockstorlek på 8 kByte medan en datalagringsprocess med 512 Byte meddelanden skulle exekvera bättre om blockstorleken var 512 Byte. Större blockstorlekar möjliggör en högre dataöverföringsgrad medan små blockstorlekar ger bättre utnyttjande av diskutrymme.

D-NIX filsystem tillåter olika blockstorlekar för filsystemet från 512 Byte till 32 kByte. Blockstorleken bestäms vid initiering av filsystemet för en speciell enhet. Definitionen av filsystemets blockstorlek och de enhetsspecifika parametrarna finns lagrade på disk och läses automatiskt in då man gör "mount" på disken.

Genom att tillåta olika blockstorlekar på olika enheter som samtidigt används i systemet erhåller man optimal prestanda och utnyttjande av disken.

Spegeldiskar: I en applikation där det är viktigt att bibehålla dataintegriteten och att data finns kvar används oftast en av två lösningar: frekvent säkerhetskopiering av disken eller en extra disk vilken är en exakt kopia (spegel) av den första.

Alla system har möjlighet till säkerhetskopiering men D-NIX tillhandahåller även möjligheten att använda spegeldiskar. För applikationen ser de två diskarna ut som en enda - D-NIX hanterar automatiskt uppdateringar, läsning o s v. Om en disk slutar fungera tar den andra över automatiskt. Att utnyttja möjligheten med spegeldisk är transparent för användaren.

Om en trasig disk byts ut kommer spegelfunktionen i D-NIX att automatisk bygga upp den nya disken. Denna uppbyggnadsprocess hanteras på kärnnivån genom att kopiera block för block från den primära disken till den nyinstallerade spegeldisken.

Spegelfunktionen kan användas med två SCSI- eller SMD-kontrollenheter och diskar. Se avsnittet *Extrafuktioner för förbättrad dataintegritet.*

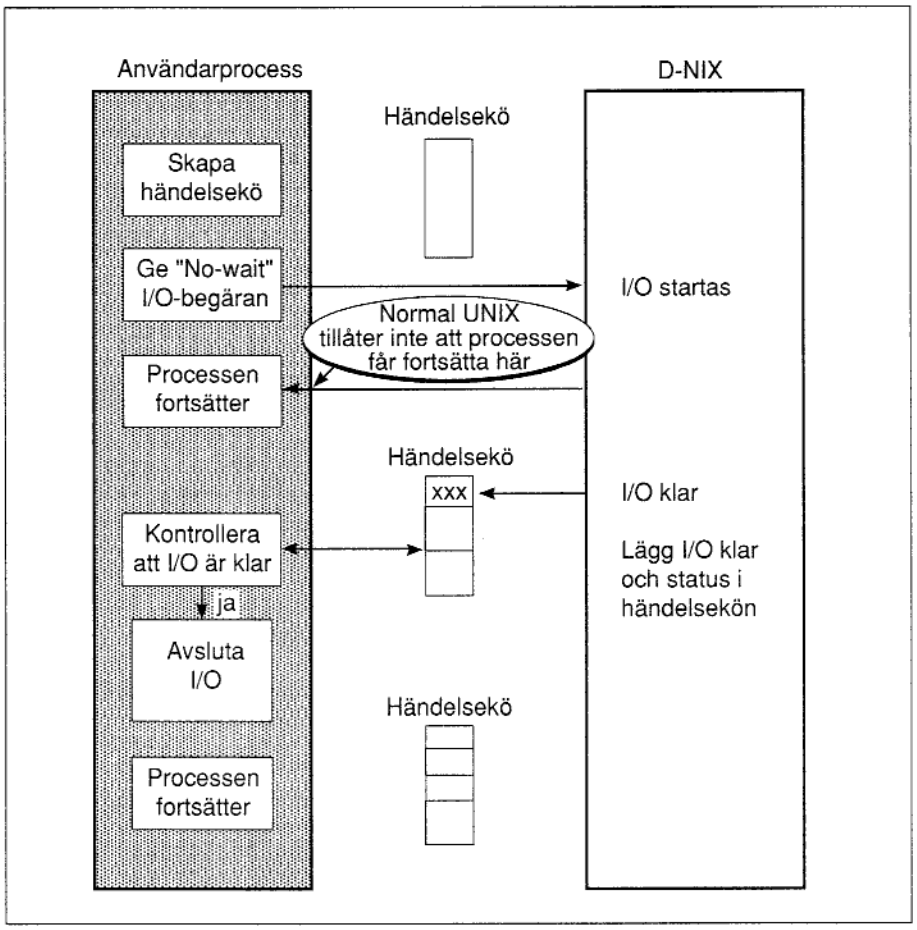
Avancerade I/O-tjänster

Asynkron I/O - ett realtidsbehov

I standard UNIX då en applikationsprocces utför en I/O läs- eller skriv beställning blir den blockerad (slutar att exekvera) tills dess att den begärda I/O-operationen är avslutad. I realtid kan flera händelser inträffa samtidigt eller mycket nära varandra i tiden. En realtidsprocess har inte råd att blockeras i väntan på att I/O-operationer avslutas eftersom en annan händelse kan inträffa vilken kräver omedelbar service.

D-NIX tillhandahåller ett sätt för applikationsprocesser att utföra asynkron I/O. I D-NIX kommer en process som utför en asynkron beställning fortsätta att exekvera då begäran har lagts in i en kö i kärna. En process som utför asynkrona I/O-beställningar kommer att kunna hantera andra händelser i väntan på att I/O-operationen avslutas. Med denna möjlighet kan applikationer definiera den tid som går åt innan en händelse kan åtgärdas.

Medan D-NIX väntar på att en asynkron I/O-beställning tas om hand, kan andra processer bearbetas.



Som vi diskuterade tidigare är de förbättringar som D-NIX tillhandahåller åtkomliga för användaren via det unika UNIX systemanropet *dnix*. Det finns två metoder för att utföra ett *dnix*-anrop: med *nowait* som är asynkront eller med *wait* som är synkront (som standard UNIX). En applikationsprocess som utför ett asynkront I/O-anrop behöver informeras då I/O-operationen är avslutad. D-NIX tillhandahåller en funktion för händelseindikering (eng. Event Notification) genom vilken applikationsprocessen informeras när I/O-operationen är avslutad.

Asynkrona anrop i D-NIX är inte endast för I/O-anrop. Snabba D-NIX semaforer och högupplösande timers är två andra exempel. I slutet på detta häft finns en lista över de systemfunktioner som stöds med systemanropet *dnix* samt visar vilka som kan användas asynkront.

Händelseindikering

Händelseindikering (eng. event notification) kan jämställas med händelseköhantering vilket är en förbättring i D-NIX. Det används för att hålla reda på alla utestående asynkrona beställningar som en process har skickat till D-NIX operativsystem. Denna funktion kombinerad med *nowait* ger användaren bra kontroll av I/O-hantering. En händelsekö används för att samla upp alla utestående beställningar. Så fort som en I/O-operation avslutas läggs resultatet av operationen in i händelsekön.

Innan en process kan börja använda asynkrona I/O-operationer måste den etablera en händelsekö. Resultatet av en operation läggs in i slutet på händelsekön av D-NIX. Händelsekön öppnas med ett av *dnix* systemanropen och kön returneras som om det vore en filpekare. För varje *nowait* systemanrop måste applikationen läsa händelsekön för att avgöra om det är klart och om det gick bra. Applikationsprocessen kontrollerar händelsekön med standard UNIX läskommandot *read(fd, buf, nbytes)*. Bufferten (buf) innehåller information om resultatet (om det gick bra eller inte) av den begärda operationen. Asynkron I/O och timerbeställningar kan avbrytas av den process som gjorde beställningen. Om detta gick bra eller inte kan också läsas ur händelsekön.

Processer i kärnan

Vid synkrona I/O-beställningar fortsätter kärnan att exekvera medan användarprocessen blockeras i väntan på att begärd operation skall avslutas. När det gäller asynkrona beställningar fortsätter användarprocessen att exekvera så fort som begäran lagts in i en kö till kärnan. När en beställning väl är köad, behandlar de flesta drivrutiner automatiskt asynkrona beställningar på drivrutinsnivå.

Asynkrona beställningar till pipe-hanteraren kan inte behandlas och köas till drivrutiner såvida det inte finns en mekanism för att fortsätta exekvera i kärnan. Med *nowait* systemanrop fortsätter applikationsprocessen att exekvera så fort begäran lagts in i kön för fil- eller pipe-hanteraren. Asynkrona D-NIX I/O-beställningar till pipe-hanteraren behandlas av speciella processer i kärnan, "kärnprocesser". För varje beställning exekveras en kärnprocess i kärn-mod under den process som gjorde beställningen. Varje kärnprocess lever lika länge som varje individuell beställning.

036/036/250/Diab Data AB

Transaktionshantering

Transaktionshantering (eng. On-Line Transaction Processing, OLTP) är en snabbt växande kategori av datorapplikationer. Snabb och tillförlitlig tillgång till databaser behövs inom sådana områden som banker, biljettreservationer, försäljningssystem och ordersystem. Ordentligt genomförd kan datoriserad transaktionshantering förbättra de flesta affärsmiljöer.

På grund av sina specifika krav använder idag de flesta OLTP-system centraliserade stordatorer med operativsystem speciellt utvecklade för detta ändamål. Sådana datorsystem är dock inte alltid den optimala lösningen för en given applikation. Den nya generationen billiga supermikrodatorer förberedda för nätverk har potential att öppna nya oräkneliga områden för tillförlitlig transaktionshantering samtidigt som de förbättrar de redan befintliga.

De flesta fleranvändarsystemen använder sig av AT&T UNIX eller något liknande som operativsystem. Olyckligtvis är den senaste versionen av System V.3 eller den planerade V.4 otillräckliga vad gäller transaktionshantering.

D-NIX tillhandahåller en egen lösning för OLTP. Följande är en lista över vad som krävs av ett operativsystem för OLTP:

- Dela data mellan alla användarprocesser (på ett nätverk)
- Tillhandahålla snabb åtkomst till databaser
- Kontrollera integriteten för applikationsdata
- Samtidigt hantera flera beställningar

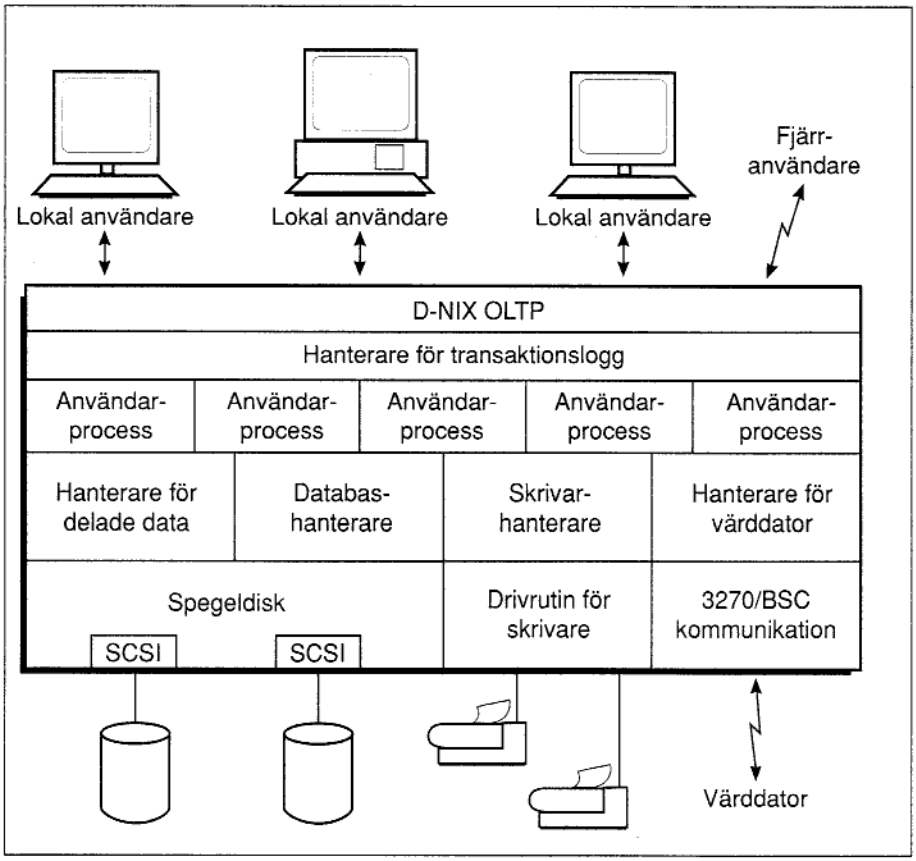
Detta kapitel diskuterar konsekvenserna av dessa OLTP-krav samt hur D-NIX tillhandahåller lösningar.

Distribuerad och transparent delning av data

I en OLTP-applikation behöver flera användare kunna dela data via filsystemet. Ett problem för standard UNIX är att tillhandahålla effektiv kontroll (åtminstone fil- och postlåsning) i hela systemet. Standard UNIX klarar av att dela data i lokala miljöer. Det klarar dock inte av distribuerade miljöer där det krävs exklusiva och delade lås på databaser, tabeller, poster och kanske till och med för olika fält. D-NIX tillhandahåller en effektiv metod för distribuerat (över nätverk) delande av data via hanterare.

Hanteraren fungerar som trafikpolis då en användarprocess vill få tillgång till data. Eftersom en hanterare ser ut som ett bibliotek eller en fil tror applikationen att den talar med UNIX filsystem. Detta gör att hanteraren isolerar datakontroll enligt systemdesignerns önskemål.

Om ett system ingår som en del i ett nätverk kan andra hanterare ge transparent funktionalitet. I D-NIX är faktiskt nätverksprogramvaran uppsatt som hanterare. Via en hierarkisk uppbyggnad (inom en eller spridd över flera hanterare) är det möjligt att få en distribuerad databasmiljö på ett nätverk.



Med hierarkisk uppbyggnad är det möjligt att få en distribuerad databasmiljö på ett nätverk.

Databasåtkomst

Databasen är den mest kritiska delen i en OLTP-applikation. Därför är systemdesignen och administrationen centrerad runt databasen. En önskvärd möjlighet är att kunna allokera kontinuerliga diskblock. Förallokering av diskutrymme ger förutbestämbara åtkomsttider och snabb dataöverföring. Det garanterar också tillgång till lagringsutrymme för transaktioner. D-NIX har möjlighet att förallokera kontinuerliga diskfiler.

Kontrollera dataintegriteten för applikationer

Om vi fortsätter med det faktum att databasen är en kritisk del i en OLTP-applikation så är det viktigt att kunna erbjuda en funktion för att kontrollera databasens integritet. Detta uppnås med hjälp av kontrollerade diskskrivningar, varierbara integritetsnivåer i filsystemet och extra funktioner för en förbättrad dataintegritet. Dessa kontrollfunktioner är:

Kontrollerade diskskrivningar: Transaktioner kräver uppdatering av informationen som lagras i databasen på en disk. Fel i maskinvaran eller programvaran som skriver till disken kan göra att innehållet i databasen inte är korrekt. Som vi diskuterade tidigare innehåller inte standard UNIX någon möjlighet att bli försäkrad om att data verkligen har skrivits till disken. D-NIX har ett systemanrop "skriv ut filbuffert" (eng. flush file buffer) med fullständig felrapportering som ser till att en transaktion verkligen skrivs ut på disken. Denna skrivfunktion fungerar både lokalt och över ett nätverk.

Varierbara integritetsnivåer i filsystemet: Snabbhet och integritet hos filsystemet är något som önskas av OLTP-applikationer. Olyckligtvis uppstår en konflikt mellan dessa två önskemål. D-NIX angriper detta problem genom att ge systemadministratören

036/036/250/Diab Data AB

möjlighet att specificera på vilken nivå det skall användas skrivminne (eng. write-caching) på filsystemets datastrukturer.

Om till exempel filhuvuden och filbibliotek inte skrivs till disken varje gång de ändras kommer filsystemet att bli mycket snabbt medans integritetsnivån blir låg då data kan förloras om systemet går ned. Om filsystemets datastrukturer tvingas ut på disk kommer filsystemets snabbhet att gå ned medans integritetsnivån går upp. Filsystemet i D-NIX kan skräddarsys för olika applikationer genom att tillåta användning av olika integritetsnivåer.

Extrafunktioner för bättre dataintegritet: On-line transaktionshantering kräver att systemet finns tillgängligt så fort en tangent trycks ned. Systemets tillgänglighet är beroende på applikationens omgivning.

I ett OLTP-system är det diskarna som har den största sannolikheten att falla ur eller gå sönder, eftersom det är den kringutrustning som används flitigast. D-NIX kan använda en funktion med spegeldiskar vilket gör det möjligt för två diskar att fungera som en. Om en disk slutar fungera tar den andra över.

Spegeldiskar ökar systemets tillförlitlighet: Två identiska fysiska diskar (en primär och en sekundär) uppfattas som en logisk disk av applikationen. Diskskrivningar utförs på båda två, och diskläsningar görs från den primära disken. Om flera läsfel uppkommer vid läsning från den primära disken kommer den sekundära att bytas ut till att vara primär.

Om ett skrivfel uppstår på någon av diskarna kommer skrivningen att göras om. Om flera skrivfel uppstår på samma disk (med samma data) kommer disken att tas ur drift.

Under förutsättning att de två diskarna har separat spänningsmatning (och separata kontrollenheter) kan en av diskarna tas bort för att repareras medans systemet fortsätter att gå.

När en disk tas tillbaka i drift kommer den automatiskt att uppdateras till en exakt kopia av den primära disken. Alla diskvolymer kan användas vid spegling, även rotvolymen.

Flera utestående begäran

OLTP-applikationer blir alltmer sofistikerade och krävande. Det är inte längre nödvändigt att begränsa applikationer till att utföra endast en sak åt gången. Detta krav är identiskt med det för realtidsapplikationer - behovet av att kunna åtgärda flera samtidiga händelser.

I realtid är svarstiderna viktiga - i OLTP är svarstiderna viktiga men inte lika krävande. För att lösa detta i D-NIX används asynkront I/O och händelser. OLTP-applikationer under D-NIX kan åtgärda flera händelser med svarstider på samma nivå som under realtid.

Clustersystem

I vissa fall kanske man finner att ett flerprocessorsystem ändå inte ger tillräcklig kapacitet. Då finns möjligheten att använda sig av kopplade system (eng. cluster). Ett DS90-CLUSTER består av en DS90 huvuddator samt en till fyra slavdatorer. Kommunikationen mellan huvuddator och slavdatorerna sker via 110 Mbit/s fiberoptiska länkar. Från huvuddatorn utgår en fiberoptisk länk till varje slavdator. Slavdatorerna kan befinna sig upp till en kilometer från huvuddatorn.

Genom sammankopplingen har följande fördelar uppnåtts:

- Ett kopplat system får betydligt större kapacitet än ett enskilt system.
- Befintliga DS90 system kan vid uppgradering till kopplade system ingå som del av det nya systemet.
- Kapaciteten på det kopplade systemet kan ökas gradvis efter behov och önskemål.
- Ingående datorer uppfattas av användaren som en enda dator, d v s samtliga i systemet ingående resurser kan nås från valfri arbetsplats.
- Ingående slavdatorer kan stannas och återstartas utan att systemet i övrigt påverkas.
- Systemets interna uppbyggnad är fullständigt transparent för samtliga applikationsprogramvaror.

Kommunikation mellan slav och huvuddator

Kommunikationen införs maskinvarumässigt genom att ett processorplan i varje slavdator bestyckas med ett speciellt processorkort som innehåller maskinvara för fiberoptisk kommunikation. Ett sådant processorkort innehåller två fiberoptiska kanaler. I slaven utnyttjas endast en kanal för kommunikation med huvuddatorn.

Huvuddatorn förses med ett eller två processorplan för optisk kommunikation beroende på om två eller flera slavar önskas. I huvuddatorn utnyttjas bägge kanalerna för kommunikation mot slavar.

Det minsta DS90-CLUSTER systemet består av en huvuddator och en slavdator. Det kan maximalt bestyckas med två processorkort för kommunikation (en i varje dator) samt sex stycken normala processorplan, tre stycken i varje dator. Detta kan jämföras med den maximala bestyckningen av att DS90 system som består av fyra normala processorplan.

DS90-CLUSTER-systemet kan utökas till två, tre eller fyra slavar. Om fler en två slavar önskas, används ytterligare ett processorplan i huvuddatorn för kommunikation med slav tre och fyra. Den maximala bestyckningen består av en huvuddator och fyra slavar, totalt sex stycken processorplan för kommunikation, två i huvudmaskinen och en i varje slav, samt 14 stycken normala processorplan, två i huvudmaskinen och tre stycken i varje slav. Tabellen nedan visar maximal processorbestyckning vid ett visst antal slavar.

036/036/250/Diab Data AB

Antal slavar	Antal kom. processorer	Max antal standard processorer
0	0	4
1	2	6
2	3	9
3	5	11
4	6	14

Antalet standard processorer växer således med antalet slavar. Dessutom, då fler slavar används fördelas I/O på flera botten processorer vilket innebär att huvuddatorn avlastas en mängd I/O-avbrott, främst gäller detta terminal-I/O.

Processnummergenerering

Vid start av systemet, tilldelas varje slav ett id, ett nummer mellan 1 och 4.

Id-nummret används för att tilldela processnummer till de processer som körs i respektive slavar. Eftersom systemet uppfattas som en enda logisk dator får en viss slav ej generera en process med ett processid som redan existerar i en annan slav eller i huvudmaskinen.

Id-nummeret används dessutom av filhanteraren för att identifiera I/O-enheter belägna i slavar.

Huvuddatorn är ansvarig för all filhantering, endast denna dator innehåller en filhanterare. Slavarna kommunicerar med huvuddatorn då fil-I/O skall utföras eller då I/O görs mot en enhet som ej är belägen i den egna slaven. Detta innebär att slavarna normalt endast innehåller en diskettenhet som används som boot-media vid start. Starten kan naturligtvis även göras från en hårddisk om sådan är tillgänglig lokalt för en slav. Efter start används filsystemet i huvuddatorn, d v s samtliga slavar "ser" exakt samma filsystem som huvuddatorn.

Systemets samtliga I/O-enheter är åtkomliga från valfri dator. Vid öppning av en enhet sänds beställningen vidare till den dator där enheten är fysiskt belägen. Om enheten befinner sig i samma dator som beställaren, exempelvis en terminalport i en slav, sker därefter I/O beställningar helt lokalt i denna dator. Detta innebär att I/O i en slav utförs utan att belasta de övriga datorerna i systemet.

Vid starten av systemet bootar huvudmaskinen upp på samma sätt som en normal DS90 maskin. Slavarna bootar upp från lokalt media, normalt diskett, och väntar på klartecken från huvudmaskin att denna är redo.

Var en användares program skall exekveras i systemet bestäms då användaren loggar in. Programmen körs i den dator där användarens inloggningsterminal är fysiskt ansluten.

Vid implementeringen har stor vikt lagts vid att en generell applikation skall kunna köras från valfri arbetsplats. Inga ändringar av programvaran krävs för att exevera dem i en clustermiljö.

Prestanda

En av målsättningen vid implementeringen har varit att program som exekveras i slavar ej skall gå synbart långsammare än om motsvarande program körs i huvudmaskinen.

Resultatet blir att fördröjningen över den optiska länken är närmast obefintlig, istället begränsas farten av hur snabbt filhanteraren kan effektuera beställningar, vilket är likvärdigt med att programmet körs i huvudmaskinen.

Lokala swap diskar

För att uppnå maximal prestanda bör man minimera antalet beställningar som sänds till huvudmaskinen. I stora system sker ofta en stor mängd minnesswapping av utvalda delar av applikationsprogrammen. Eftersom rootdisken i huvudmaskinen används som swap area innebär detta att slavar kan sända en mängd swapbeställningar över kommunikations länken.

För att råda bot på denna extra trafik har möjlighet till swappning mot lokla diskar i slavarna införts.

036/036/250/Diab Data AB

Programspråk

D-CC C-kompilator

D-CC är en optimerande C-kompilator som genererar mycket snabb och kompakt kod. D-CC använder sig av ett flertal optimeringsmetoder som normalt endast används av stordatorkompilatorer. Algoritmen för allokering av register är mycket avancerad. I de fall där det uppstår en valmöjlighet mellan snabbhet och kodstorlek prioriteras snabbhet. Benchmark-tester har visat att D-CC är en av de snabbaste kompilatorerna på marknaden.

D-CC innehåller en komplett implementering av C-språket som det är definierat i "The C Programming Language" skriven av Kernighan och Ritchie. D-CC är även en fullständig implementering av förslaget till ANSI-standard för C (X3.J11). I de fall där frågeställningar dyker upp vad gäller implementeringen av olika funktioner i C-språket används Microsoft/AT&T implementeringen av PCC (Portable C Compiler) som vägledning.

ACE/FORTRAN

ACE/FORTRAN ger användaren tillgång till en uppdaterad version av UNIX F77 kompilatorn. Den ger en fullständig implementering av FORTRAN 77 som det är specificerat i FORTRAN ANSI X3.9-1978 standarden.

ACE/PASCAL

Ger användaren tillgång till en väl fungerande och testad Pascal. Språket ansluter sig till den från ISO föreslagna standarden och innehåller även de bästa bitarna från Pascal/MT+, t ex dynamisk stränghantering.

RM/COBOL

RM/COBOL omfattar de mest använda möjligheterna i COBOL för att utveckla affärsapplikationer. Det är baserat på ANSI-standarden från 1974 och 1985 samt godkänd av GSA. Språket stöder åtkomstmetoder på nivå två av sekventiella, relativa och indexerade filer. Det finns även kraftfullt stöd för skärmhantering och låsning av filer och poster för fleranvändaråtkomst av filer.

RM/COBOL Runtime

Programvaran RM/COBOL RUNTIME gör det möjligt att exekvera tidigare kompilerade RM/COBOL-program under kontroll av D-NIX. Run-time modulerna laddar och exekverar objektprogrammen samt allokerar lagringsutrymme och filbuffertar.

D-BASIC V och BasC

Två programpaket finns tillgängliga för användare av Basic. D-BASIC V är en intepreterande kompilator för språket Basic. Till skillnad från de flesta Basic-system som tolkar källkoden vid exekvering så tolkar och kompilerar D-BASIC V koden då den skrivs in. Den främjer även välstrukturerad programmering via flerradiga procedurer och funktioner samt genom automatisk indentering av slingor.

BasC tar ett D-BASIC V program och gör om det till C-kod. Sedan kompileras och länkas programmet med ett annat bibliotek speciellt avsett för D-BASIC V. Med hjälp av BasC kan ett kompilerat D-BASIC V program exekvera upp till 50 gånger snabbare.

Båda Basic-paketen är anpassade till ANSI X3.60-78 standarden och innehåller ytterligare funktioner för åtkomst till ISAM databaser och UNIX interprocesskommunikation.

Assembler

Det finns tillgång till en assembler för program skrivna med standardsyntax för Motorola MC68000 mikroprocessorfamiljen.

036/036/250/Diab Data AB

Tillvalsfunktioner

TTY-gränssnitt med hög prestanda

Terminalkommunikation i UNIX har en hög overhead och med ett ökat antal användare så minskar systemprestanda. En förbättring av DS90 lösning innehåller en intelligent terminalkoncentrator som ger högre prestanda än konventionella terminalgränssnitt. Terminalkoncentratorn är ett intelligent VME-kort som stöder 10 portar med en total bandbredd på 90 kByte/sekund. Varje port kan arbeta med en hastighet på upp till 38400 Baud.

Konvetionella TTY-gränssnitt överför ett tecken (en Byte) åt gången. Processorn måste användas för vart och ett, d.v.s den avbryts efter varje läsning eller skrivning av ett tecken. Detta skapar hög overhead. Med DS90 terminalkoncentrator hanteras all tecken I/O på koncentratorns kort.

Gränssnittet mellan terminalkoncentratorn och processorn sker via en FIFO. För en skrivning till terminalen skickas så många tecken som möjligt till FIFO'n med en enda överföringsskur. T.ex vid en läsning av ett UNIX-kommando kommer terminalkoncentratorn att läsa hela raden och avbryter processorn först då användaren tryckt på returtangenten. Detta förbättrar avsevärt prestandan vid terminalkommunikation (RS232).

Stöd för optiska skivminnen

D-NIX stöder ett filsystem för optiska diskar av WORM-typ (Write Once Read Many) via en speciell filhanterare. Filsystemet stöder vanliga filer och bibliotek. En vanlig fil med data kan delas upp och utökas men inte förändras. Ändringar av data i filen kommer att skapa en ny generation av filen. Filsystemet för WORM-diskar uppträder på samma sätt som ett standard UNIX-filsystem med de ovan nämnda skillnaderna. Gamla generationer av en fil kan hämtas tillbaka eftersom data på disken aldrig förstörs. Vidare kan borttagna filer hämtas tillbaka då filsystemet kan återskapa disken så som den såg ut vid en bestämd tidpunkt.

Stöd för band med hög lagringskapacitet

DS90-systemet stöder band med hög lagringskapacitet (2.2 GByte) för generell lagring eller säkerhetskopiering.

D-MENU säkert användargränssnitt

Användargränssnittet i D-MENU ger en användarvänlig åtkomst till DS90-systemen samt ytterligare säkerhet. Menyer kan vara av standardtyp för systemet eller användarspecifika. D-MENU gör det möjligt att ytterligare förbättra säkerheten vid åtkomst av terminaler och till skrivare.

När en användare loggar in kommer det upp en meny. Denna meny kan vara antingen en standarmeny eller användarspecifik. D-MENU ger möjlighet till en flexibel arkitektur av menyer. En meny används vid inloggning och från denna kan användaren gå vidare via undermenyer. Alla D-NIX kommandon eller applikationer kan användas via en meny.

D-MENU ger tillgång till yterligare en säkerhetsnivå utöver det som D-NIX erbjuder. Med D-MENU kan användare begränsas till att använda endastvissa utvalda terminaler.

Fel som uppkommer vid användningen av D-MENU presenteras för användaren på ett lättförståligt sätt. Detta inkluderar fel i menyhanteraren och fel som uppkommer i program och vid systemanrop som görs från menysystemet.

D-MENU innehåller en menyeditor. Den gör det enkelt för den som utvecklar att skapa och uppdatera menyer. Både menyhanteraren och editorn är språkoberoende. Detta gäller alla texter i systemet, alla felmeddelanden, dialogtexter och hjälpfiler.

D-NIX Utbyggnadspaket/SÄK1

D-NIX Utbyggnadspaket/SÄK1 är ett komplement till D-NIX och D-MENU när man önskar en utökad säkerhet i ett DS90 system. Paketet ligger som ett skal utanpå den vanliga behörighetskontrollen. Det innebär att det inte finns några möjligheter att på obehörigt sätt kringå skalet för att komma åt operativsystemet.

SÄK1 ger tillsammans med D-MENU långtgående kontroll över bl a:

- Styrning av inloggning till vissa tider
- Automatisk tidsstyrd utloggning efter viss tid av inaktivitet
- Motringning vid modemuppkoppling
- Systemadministration endast från konsolarbetsplatsen

D-NIX Utbyggnadspaket/SÄK2

Vill man gå ytterligare ett steg på vägen mot ett säkert system kan D-NIX Utbyggnadspaket/SÄK2 användas. Detta bygger på AT&T UNIX System V/MLS, som är en säkerhetsmässigt förbättrad version av operativsystemet UNIX System V. Det är utvecklat för att möta det amerikanska försvarets säkerhetsklassificeringen för system med flera säkerhetsnivåer (B1). Operativsystemet är helt kompatibelt med UNIX System V.

SÄK2 ger möjligheter för användaren att fördela information i en rad olika behörighetsnivåer och projektkategorier. Totalt finns 1024 kategori- eller projektklasser, var och en med 255 sekretessnivåer. Även processer, meddelandeköer, minnespartitioner, terminal- och skrivarutgångar etc kan klassificeras i sekretessklasser. Alla objekt (filer) etiketteras automatiskt vid skapandet enligt det privilegium som den skapande processen har.

SÄK2 utnyttjar en modell för tvingande åtkomstkontroll som innebär att man inte kan läsa filer i överliggande sekretessnivåer och inte heller kan skriva till filer i underliggande nivåer. Denna modell åstadkoms genom att både subjekt (processer m m) och objekt (filer m m) automatiskt förses med etiketter där säkerhetsklassningen kodas in. Vid varje åtkomstförsök kontrolleras dessa etiketter mot varandra.

Systemet har även en omfattande säkerhetslogg där varje åtkomst, försök till åtkomst och genomförd aktivitet registreras. I säkerhetsloggen kan upp till 20 valbara händelsetyper registreras. Informationen lagras i komprimerat, binärt format på fil. Särskilda åtgärder finns i systemet som säkerställer att åtkomsthistoriken inte går förlorad och som dessutom omöjliggör användning av systemet om funktionen saknas.

I SÄK2 finns även en styrd lösenordsfunktion som ger användaren slumpvist alernativa lösenord att välja från.

036/036/250/Diab Data AB

Anteckningar

Anteckningar

036/036/250/Diab Data AB