Systemadministration

Handbok Systemadministration

Uppdaterad för D-NIX 5.2 release 1.2

9821-10

DATAINDUSTRIER DIAB AB

Box 2029 S-183 02 TÄBY
SWEDEN

Revisionsinformation

Revisionsinformation

Revision	Datum	Kommentar
A	870410	Första versionen av dokumentationen

Innehållsförteckning

Innehållsförteckning

INLEDNING

Välkommen till denna handbok som behandlar administrationen av datorsystem ur DIAB Series 90. Handboken är avsedd att hjälpa dig som är ansvarig för driften av systemet.

Systemdokumentation:

```
9764-10 Användarhandbok D-NIX
97XX-10 Installationshandbok DS90-YY
9710-10 Bok:  UNIX-handboken
9701-10 D-MENU Menysystem
9830-10 DS90 Systempärm
```

I denna handbok refereras ibland till någon av ovanstående handböcker. Vid dessa tillfällen används följande förkortningar:

```
AH  Användarhandbok D-NIX
IH  Installationshandbok DS90-YY
DM  D-MENU Menysystem
SP  DS90 Systempärm
PH  D-NIX Programmer Manual (extra)
```

Vi är mycket tacksamma för kommentarer till denna handbok från dig som användare. Har du ideer om hur den skulle kunna förbättras, så använd gärna det svarskort som finns längst bak i handboken.

Lycka till med ditt nya datorsystem.

Dataindustrier DIAB AB
Täby, 1987.

1

2

2. Systemhandhavande

2.1 Startförfarande

2.1.1 Manuell start

ALLMÄNT

Normalt startas systemet helt automatiskt. Det finns emellertid tillfällen då man behöver starta systemet manuellt t ex

* vid formattering av minnesmedia.

* vid ändring av programladdningsparametrar (bootstrap),

* vid problem med automatisk uppstart.

Vid manuell start kan uppstart och programladdning styras stegvis. Starten sker på följande sätt och förutsätter att systemet är helt avstängt. Manuell start kan även ske efter fullständig nedtagning av systemet med kommandot **'/etc/shutdown -k'** utan att spänningen slagits av.

OBS! Terminalen ska ge signalen DTR enligt kapitel 16 och huvudkonsolen skall vara inkopplad med standard kommunikationsparametrar enligt **IH**.

För alla modeller, utom DS90-11: Manuell start förutsätter att nyckeln på datorn står i läget OFF. För att starta vrids nyckeln först till RESET och därefter direkt tillbaka till OFF. Spänningen slås då på automatiskt och uppstarten sker i manuell mod.

För DS90-11 enbart: Manuell start förutsätter att nyckeln på datorn står i läget MANUAL. Starten sker direkt när spänningen slås till med huvudbrytaren.

Manuell start sker genom fyra bootnivåer:

Bootnivå 0: Bootstrap PROM styr inladdningen av ett laddningsprogram från skivminne eller flexskiva. En kort text 'M680x0 prom' skrivs ut på huvudkonsolen och operatören svarar med <RETURN> eller namnet på det minnesmedium där laddningsprogrammet finns lagrat. Efter inladdning anger operatören namnet på den enhet där filsystemet finns.

Bootnivå 1: Laddningsprogrammet (bootstrapsystemet) är i drift. Detta är ett enkelt skivoperativsystem och vissa, korta program kan användas på denna nivå. Laddningsprogrammet kallas ibland även för 'stand-alone-systemet'. Prompten på denna nivå är >. Med kommandot **dnix** fortsätter startsekvensen till nästa nivå efter laddning av D-NIX.

Bootnivå 2: Enanvändarmod. Super-user är ensam i systemet. Prompten är #. Kommandot **telinit 2** ger start upp till den sista nivån.

Bootnivå 3: Fleranvändarmod. Systemet kräver inloggning. Prompten efter inloggning är ¤ för normalanvändare och # för superuser. Exakt vad som sker vid övergång till bootnivå 3 (fleranvändarmod) styrs av innehållet i filen /etc/inittab.

BOOTNIVÅ 0

Direkt efter <RESET> eller spänningstillslag utförs en hårdvarutest och kontroll av att erforderlig minneskapacitet finns tillgänglig. När detta är genomfört svarar systemet

```
M680XX prom        (XX = 00 eller 20)
```

och avvaktar operatörens kommando vid huvudterminalen.

Operatören skall nu ange den enhet varifrån laddningsprogrammet skall läsas in.

Om operatören svarar med enbart <RETURN> tas laddningsprogrammet från den standardenhet som finns angiven i datorns NVRAM. Normalt värde är det interna winchesterminnet. I detta fall läses laddningsprogrammet in och startas direkt i bootnivå 1 enligt nästa avsnitt.

Operatören kan istället uttryckligen ange enheten varifrån programmet skall läsas in. Enhetsnamnet avslutas med <RETURN>. Följande enhetsnamn gäller för de olika modellerna:

DS90 -	10/10E	11	20/21
Intern winchester	si(16,0)	si(0,0)	si(16,0)
5 1/4 tum diskett 720 kbyte	mf(64,0)	mf(64,0)	--
5 1/4 tum diskett 1,2 Mbyte	mf(8,0)	mf(8,0)	mf(8,0)

I enhetsnamnet anger namnet, t ex **si**, enheten och första talet ett kanalnummer. För laddning från den BOOT-diskett som medföljer systemet (1.2 Mb), skall **mf(8,0)** anges, (720 kb) skall **mf(64,0)** anges. För ytterligare information om dessa kanalnummer, se avsnitt 15.

Därefter läses laddningsprogrammet in från den angivna enheten och startas, Programmet frågar först efter den enhet som skall användas som "root"-filsystem.

```
Root device "xx(nn,nn)" ? _
```

Här anger operatören samma eller annan enhet enligt listan ovan, varefter laddningsprogrammet startar. Om laddningsprogrammet lästs in från disketten men winchesterenheten angivits som root-device bör nu disketten tas ut.

BOOTNIVÅ 1 Laddningsprogrammet

Laddningsprogrammet (Stand-Alone-Systemet) är i drift. Det är ett enkelt operativsystem och vissa, enkla program kan köras. Prompten > visar att kommandon kan ges.

I det fall där systemet inte har stängts av korrekt ges en varningstext om att filsystemet inte är korrekt. Filsystemet skall, om detta inträffar, korrigeras med kommandot **fsck**, som utför test av filsystemet på det fasta skivminnet i massminnesenheten. Kommandot **fsck** får endast användas i laddningsprogrammet (bootnivå 1), om den enhet som är root-filsystem ska testas. Kommandot ges på följande sätt:

```
/etc/fsck -rr si(xx,x)
```

xx,x har följande värden på de olika modellerna:

DS90 -	10/10E	11	20/21
Intern winchester	si(16,0)	si(0,0)	si(16,0)

I biblioteket **/sas** på den boot-diskett som levereras med systemet finns några program som endast kan användas på denna nivå, bl a de nedan nämnda. Där finns även vissa testprogram som beskrivs i avsnitt 8 och som måste köras på denna nivå. Namnet /sas är en förkortning av "stand alone system", vilket innebär att endast ett program i taget kan köras och med en enda användare.

Exempel:

/sas/bootpar	ändrar laddningsparametrar i NVRAM. Se även /etc/bootpar.
/etc/fsck	används på denna nivå för att testa och korrigera root-filsystem.

Avsnitt 3 innehåller en detaljerad beskrivning över "stand-alone"-systemet.

Om systemet är i bootnivå 0 eller 1 och ingen skrivning/läsning sker på skivminne eller diskett, kan systemet återstartas genom att nyckeln förs till läget RESET och tillbaka till OFF (MANUAL) eller AUTO. Med samma villkor kan även spänningen slås av med huvudbrytaren baktill på datorsystemet. Efter att operativsystemet startats enligt nedan, måste däremot en korrekt avstängning ske enligt kapitel 2.3.

Operativsystemet laddas in och startas med kommandot **dnix**.

```
>dnix
```

Alternativt kan en valfri sträng anges efter kommandot, varvid standard systemparametrar används istället för de som angivits av kommandot **/etc/mkcfig** (kapitel 5). Operativsystemet startas då med '**dnix** xxx'. Den angivna strängen kan bestå av vilka tecken som helst.

Vid systemstart testar D-NIX minnet och skriver sedan ett systemmeddelande med revisionsnummer och den interna minneskapaciteten uttryckt i kbytes.

```
OS:INFO: D-NIX X.X Virtual Ver X.X -
         System xxxk  User xxxxk  Swap x.xM
```

Därefter startar en initieringsprocess från filen /etc/init som initierar operativsystemet. Denna initieringsprocess 'init' är process 1 och stannar i systemet som 'urfadern' till alla övriga processer.

BOOTNIVÅ 2 Enanvändarmod (systemnivå s)

I bootnivå 2 är operativsystemet i enanvändarmod. Enbart huvudterminalen kan användas och användaren är privilegierad (super-user) och behöver alltså inte logga in. **Prompten #** visar att kommandon kan ges och att alla kommandon kan användas. Man kan även logga in som normalanvändare med kommandot **su** utan att lämna enanvändarnivån. Med **su** behöver superusern inte känna till användarens lösenord utan kan logga in direkt. I bootnivå 2 kan alltså flera processer startas men med endast en användare i systemet.

I enanvändarmod används alltid enheten **/dev/syscon** som huvudkonsol och terminalparametrarna sätts upp enligt filen /etc/ioctl.syscon. Filen /etc/inittab används inte i enanvändarmoden. Se kapitel 4 för detaljer. Denna mod kallas för systemnivå 'S'.

Notera att en återgång till enanvändarnivån kan ske från den normala fleranvändarnivån (bootnivå 3) med kommandot "**/etc/shutdown**". Se kapitel 2.3.

Övergång till fleranvändarmod sker normalt med kommandot:

```
telinit 2
```

Om systemet tidigare varit i fleranvändarmod, kan enanvändarmoden avslutas genom utloggning (CTRL-D) som svar på prompten #. I detta fall ges en fråga på huvudterminalen efter en ny systemnivå (run-level):

```
ENTER RUN-LEVEL (0-6,s or S):_                (Svara: 2)
```

Denna fråga ska besvaras med **2**, för att starta i fleranvändarmod. Se nästa sida samt /etc/inittab i kapitel 4 för detaljer.

Kommandorutinen i enanvändarmod avslutas och systemet övergår till fleranvändarmod.

OBS! Med enbart BOOT-disketten som root-filsystem, kan fleranvändarmod inte användas, eftersom alla systemfiler som behövs för detta inte ryms på den BOOT-diskett som medföljer systemet.

BOOTNIVÅ 3 Fleranvändarmod

Vid övergång till fleranvändarmod läser 'init'-processen filen **/etc/inittab** och startar upp de kommandon som anges där. Kommandon exekveras bl a för att rensa bort gamla temporärfiler och starta vissa systemprocesser, t ex nätverksprocesser. Här startas bland annat en kommandoavkodare som utför kommandona i filen **/etc/rc**. En login process (/etc/getty) startas för varje ansluten terminal som skall vara aktiv.

Alla numeriska systemnivåer (0..6) inom operativsystemet är fleranvändarmoder, men som standard används systemnivå 2, enligt de definitioner som ligger i filen /etc/inittab. Systemnivå 0 är reserverad. I /etc/inittab får inga processer vara aktiva på systemnivå 0.

I filen /etc/rc finns systemkommandon som alltid utförs vid övergång till fleranvändarmod. (Se /etc/rc i kapitel 4). Bl a ger dessa ett systemmeddelande till huvudterminalen, t ex:

```
DataIndustrier DIAB AB D-NIX 5.X Y.Y Virtual
```

Bootnivå 3 är den nivå där systemet är avsett att normalt arbeta dvs som ett fleranvändarsystem. Fleranvändarmoden kräver inloggning.

Normalt sker automatisk start direkt till denna bootnivå som normalt anges i NVRAM (parametern 'boot level'). Vid automatisk start kan nyckeln tas ut i läge AUTO, d v s rakt upp, så att systemet alltid startar i denna bootnivå.

I samband med inloggningen kan ytterligare en systemkonfiguration utföras. Före inloggning sätts först baudrate m m, systemets **nodnamn** visas på terminalen, följt av en systemtext ur filen **/etc/issue**. Frågetexten vid inloggning tas ur filen **/etc/gettydefs** och under inmatning av användarnamnet kan vissa terminalparametrar ändras automatiskt.

Efter godkänd inloggning bearbetas först automatiskt kommandon i **/etc/profile** följt av kommandon i filen **.profile** i användarens eget bibliotek. Via **/etc/profile** visas bl a eventuellt systemmeddelande från filen **/etc/motd**. Därefter visas prompten ¤ eller # och systemet väntar på operatörens kommandon.

Det är även möjligt att i förväg bestämma att en användare automatiskt enbart skall ha tillgång till ett enda program. Detta anges i så fall i filen **/etc/passwd** som sista parameter. Detta program startas då direkt istället för att någon prompt ges. Se vidare avsnitt 4.

Under bootnivå 2 och 3 refereras skivminne, diskettenhet och kassettstreamer enligt **namn (major,minor)**, och följande värden gäller för de olika modellerna. Se även kapitel 15.

DS90 -	10/10E	11	20/21
Skivminne	si0(5,16)	si0(5,0)	si0(5,0)
Diskettenhet	mf0(9,8)	mf0(9,8)	mf0(9,8)
Kassettstreamer	st0(5,128)	st0(5,144)	mf0(5,128)

2.1.2 Automatisk start

Den automatiska starten sker genom att operativsystemet laddas in från winchesterminnet enligt hela den sekvens som gäller för manuell start. Den enda skillnaden är egentligen att allt detta sker utan operatörens medverkan. Starten i AUTO-läget sker till den bootnivå (0,1,2 eller 3) som finns inlagd i ett NVRAM (ett fast programmerbart minne) i systemet. Denna bootnivå kan ändras, liksom definitionen av den enhet från vilken start skall ske. Se vidare avsnitt 3, kommandot **/sas/bootpar**.

Starten förutsätter att systemet fullständig stängts av på normalt sätt eller tagits ner fullständigt med kommandot **/etc/shutdown -k**.

För alla modeller, utom DS90-11: Normalt startas systemet genom att vrida nyckeln från läget OFF till AUTO. varefter användaren loggar in med namn och lösenord. Automatisk start sker även om nyckeln är i AUTO då huvudbrytaren baktill slås på.

För DS90-11 enbart: Normalt startas systemet med nyckeln i läget AUTO. Starten sker direkt när spänningen slås till med huvudbrytaren.

När nyckeln lämnas i AUTO-läget efter en systemavstängning, och huvudströmbrytaren ej slagits av, påbörjas automatiskt ny start ca 10 sekunder efter att systemet helt stängts av. Att avstängningen är klar visas med ledtexten:

```
OS INFO: System halted
```

När nyckeln på datorn ställs i läge AUTO, skyddas systemet och inloggning krävs alltid. Detta förutsätter dock att startparametern 'boot level' i NVRAM är bootnivå 3. Nyckeln kan i detta läge tas ut.

2.2 Hantering av massminnen

2.2.1 Formattering av disketter på fleranvändarnivå /etc/format

Formattering av disketter sker med programmet **/etc/format**. Det kan användas i både en- och fleranvändarmod när man är inloggad i systemet.

5 1/4 tums disketter

För att formattera en 720 Kbytes 5 1/4 tums diskett startar man programmet, och dialogen kan se ut på följande sätt:

```
/etc/format <RETURN>
Disk format program
Enter device: /dev/mf0
Format device mf, unit 0 - OK? y

Selected mini floppy parameters
Number heads            2
Number cylinders       80
Number sectors/track    9
Sector size           512
Interleave factor       1
Parameters OK? y
Formatting disk...
Disk formatted
```

/etc/format	Kommandot självt.
Disk format program	Formatteringsprogram.
Enter device:	Ange den enhet som skall formatteras, i detta fall /dev/mf0 mf = 5 1/4 tums diskettenhet
Parameters OK? y	Start av formatteringen.
Formatting disk...	Skivan formatteras.
Disk formatted	Skivan är formatterad.

I listan som skrivs ut visas standardparametrarna, men användaren kan ändra dessa vid behov innan formatteringen sker. Formatteringen kan även startas direkt om standardparametrarna ska användas. Det görs med kommandot:

```
/etc/format /dev/mf0
```

För att formattera en "high density" 1.2 Mbytes minidiskett måste antal sektorer/spår ändras till **15** genom att svara **n** på frågan "Parameters OK?" och ange det ändrade parametervärdet vid frågan "Number sectors/track", eller via direktkommandot:

```
/etc/format -r15 /dev/mf0
```

För mer information om formatet hos disketter se avsnitt 15 i denna handbok, se även **AH**.

8 tums disketter

För att formattera en 1 Mbytes 8 tums diskett startar man programmet, och dialogen kan se ut på följande sätt:

```
/etc/format <RETURN>
Disk format program
Enter device: /dev/sf0
Format device sf, unit 0 - OK? y

Selected mini floppy parameters
Number heads           2
Number cylinders      77
Number sectors/track  26
Sector size          256
Interleave factor      1
Parameters OK? y
Formatting disk...
Disk formatted
```

/etc/format	Kommandot självt.
Disk format program	Formatteringsprogram.
Enter device:	Ange den enhet som skall formatteras, i detta fall /dev/sf0 sf = 8 tums diskettenhet
Parameters OK? y	Start av formatteringen.
Formatting disk...	Skivan formatteras.
Disk formatted	Skivan är formatterad.

I listan som skrivs ut visas standardparametrarna, men användaren kan ändra dessa vid behov innan formatteringen sker. Formatteringen kan även startas direkt om standardparametrarna ska användas. Det görs med kommandot:

```
/etc/format /dev/sf0
```

För mer information om formatet hos disketter se avsnitt 15 i denna handbok, samt den release notice som medföljer alla leveranser av operativsystemet. Se även **AH**.

2.2.2 Initiering av filsystem på diskett - mkfs

Innan en tom diskett eller ett tomt skivminne kan användas som ett filsystem måste ett filsystem skapas (initieras) på enheten med **/etc/mkfs**. Samtidigt utförs en test av hela disketten/skivminnet. Initieringen förutsätter att formattering är gjord. Den interna winchesterenheten är alltid färdiginitierad vid leverans och innehåller alltid ett grundpaket av operativsystemet.

Initieringen utförs med kommandot **/etc/mkfs**. Disketten rensas helt och ett nytt filsystem skapas med ett tomt root-bibliotek. Det är först efter det att ett filsystem är initierat som det är möjligt att lägga till, dvs använda kommandot **mount**, och lagra filer i filsystemet.

Initiering av filsystem på 5 1/4 tums diskett

```
/etc/mkfs  /dev/mf0
```

```
etc          Kommandot är placerat i detta bibliotek.
mkfs         Kommandot självt.
/dev/mf0     Namnet på den diskettenhet som skall ini-
             tieras.
```

På skärmen kommer information att skrivas ut under initieringen. När initieringen är slutförd skrivs prompten # ut.

Volymstorleken känns av automatiskt av 'mkfs', men kan liksom blockstorleken (default 2 kb) väljas manuellt. För detaljer, se kommandot **mkfs** i **AH**.

OBS! Disketter som enbart skall användas för att lagra backupfiler med kommandot **tar** behöver inte initieras. De måste däremot vara formatterade.

Initiering av filsystem på 8 tums diskett

```
/etc/mkfs  /dev/sf0
```

```
etc          Kommandot är placerat i detta bibliotek.
mkfs         Kommandot självt.
/dev/sf0     Namnet på den diskettenhet som skall ini-
             tieras.
```

På skärmen kommer information att skrivas ut under initieringen. När initieringen är slutförd skrivs prompten # ut.

Volymstorleken känns av automatiskt av 'mkfs', men kan liksom blockstorleken (default 2 kb) väljas manuellt. För detaljer, se kommandot **mkfs** i **AH**.

OBS! Disketter som enbart skall användas för att lagra backupfiler med kommandot **tar** behöver inte initieras. De måste däremot vara formatterade

2.2.3 Lägga till/Ta bort filsystem - mount, umount

Normalt finns alla systemprogram och systemfiler på winchesterenheten men i en del fall måste andra media kunna läsas och skrivas t ex en 5 1/4 tums diskett. Det nya filsystemet kopplas till ett bibliotek i gamla filsystemet med kommandot **mount**. Biblioteket måste redan existera som bibliotek under root-biblioteket och det måste vara tomt. Användaren måste ha både läs- och exekveringsprivilegier dvs. r och x privilegier, i biblioteket. Dessa privilegier kommer att ändras till det nya filsystemets privilegier i och med att **mount** har utförts.

Det nya filsystemet måste existera på den enhet eller fil som ska läggas till. Jämför kommandot **mkfs**.

Filsystemet tas bort från root-filsystemet med kommandot **umount**, varvid filsystemet uppdateras och kopplas bort korrekt.

OBS! umount-kommandot måste användas innan disketten tas ut.

I D-NIX, till skillnad mot Unix, kan det nya filsystemet använda andra filhanterare (t.ex för ABC- eller MS-DOS filsystem) än de som används i rootsystemet.

Tillägg av ett D-NIX filsystem på 5 1/4 tums diskett

Disketten sätts in i diskettenheten på systemet. Användaren måste ha superuserprivilegier och vara initierad.

```
/etc/mount  /dev/mf0  /mf0
/etc/mount  /dev/si32  /si32
```

/etc	Kommandot är placerat i detta bibliotek.
mount	Kommandot självt.
/dev/mf0	Fysiska drivprogrammet för interna 5 1/4 tums diskettenheten i systemet.
/mf0	Namn på kopplingsbiblioteket. Detta är ett vanligt bibliotek och ska vara tomt. Det är via detta bibliotek som systemet har tillgång till nya filsystemet.

När all hantering med disketten är klar måste kommandot **umount** användas innan disketten får tas ut ur diskettenheten.

om mount ger felet:
can not get exclusive write on mounttab
ska filen i /tmp tas bort.

Tillägg av ett D-NIX filsystem på 8 tums diskett

Disketten sätts in i diskettenheten på systemet. Användaren måste ha superuserprivilegier och vara initierad.

```
/etc/mount  /dev/sf0  /sf0
```

/etc	Kommandot är placerat i detta bibliotek.
mount	Kommandot självt.
/dev/sf0	Fysiska drivprogrammet för interna 8 tums diskettenheten i systemet.
/sf0	Namn på kopplingsbiblioteket. Detta är ett vanligt bibliotek och ska vara tomt. Det är via detta bibliotek som systemet har tillgång till nya filsystemet.

När all hantering med disketten är klar måste kommandot **umount** användas innan disketten får tas ut ur diskettenheten.

Tillägg av ett D-NIX filsystem på yttre skivminne

Ett eller flera yttre winchesterskivminne kan kopplas till root-filsystemet på samma sätt som en diskett men med andra mount-parametrar. Förutsättningen är att ett filsystem har skapats med **mkfs** på det externa skivminnet.

För detaljer hänvisas till dokumentation medföljande det externa skivminnet.

Exempel på hur ett filsystem tas bort

Innan en diskett avlägsnas ur diskettenheten måste systemet stänga det externa filsystemet korrekt, vilket innebär att eventuella filbuffrar skriv ner på disketten och filsystemet kopplas bort från root-filsystemet.

Samma sak gäller om externa större skivminnen används.

För att ta bort en "mountad" diskett ur root-filsystemet ges kommandot:

```
/etc/umount  /dev/mf0
```

/etc	Kommandot är placerat i detta bibliotek.
umount	Kommandot självt.
/dev/mf0	Fysiska drivprogrammet för den interna 5 1/4 tums diskettenheten i systemet.

Efter att **umount** kommandot använts kan disketten tas ur diskettenheten.

2.2.4 Initiering av fast skivminne

Följande beskrivning förklarar vad som behöver göras vid initieringen av ett nytt internt skivminne som rootsystem från distributionsdisketterna (BOOT-disketten).

OBS! Vid leverans av ett system är det fasta interna skivminnet alltid färdiginstallerat med systemprogram.

OBS! Denna beskrivning förutsätter att skivminnet är formatterat från leverantören. Formattera emellertid aldrig ett skivminne utan att ta kontakt med din leverantör. Normalt förstörs inte formatteringen om något fel skulle uppstå. Det är endast när läs/skrivhuvuden eller någon annan maskinvara gått sönder som en formattering kan behövas. I dessa fall krävs att en servicetekniker utför reparationen.

För att skapa ett nytt filsystem på ett fast skivminne ska systemet först startas i manuell mod upp till enanvändarmod (Bootnivå 2) med disketten som root-device. Disketten som ska användas är märkt:

Exempel:

```
D-NIX 5.2   079-87XX-XX/ x(x)
Version X.xx
BOOT DS90-XX
mount format / hd 1.2 Mb
DataIndustrier DIAB AB
```

Bildkärmen bör efter uppstart se ut så här:

```
M680XX prom                (XX = 00 eller 20)
mf(8,0)
Root device "xx(nn,nn)" ? mf(8,0)
> dnix
OS:INFO: D-NIX X.X Virtual Ver X.X -
         System xxxk  User xxxxk  Swap x.xM
#
```

Stegen i det följande är:

1. Skapa ett filsystem på det fasta skivminnet med **/etc/mkfs**.

Ett filsystem skapas på skivminnet med kommandot **/etc/mkfs**. Systemet är startat manuellt från BOOT-disketten upp till bootnivå 1 (enanvändarmod).

Exempel: — Se Systemanteckningar.

```
/etc/mkfs  /dev/si0
           /dev/si32
```

'**mkfs**'-kommandot känner själv av volymsstorleken och använder som standard blockstorlek 2 kbyte.

Filsystemet på skivminnet är nu initierat men innehåller ännu inga filer. Interna skivminnet används normalt som root-filsystem, varför systemfilerna nu måste kopieras över och vissa specialfiler skapas. Om ett yttre skivminne initierats på detta sätt kan det nu direkt användas genom att kopplas in i root-filsystemet som ett underbibliotek med kommandot **mount**.

När filsystemet på skivminnet är initierat skall programmet **harddoit** köras för att överföra data från BOOT-disketten till skivminnet samt initiera de enhetstabeller som krävs för att skivminnet ska kunna användas som root-filsystem.

OBS! I detta läge bör emellertid alltid den Release Notice som medföljer systemet läsas. Där kan eventuella avvikelser från nedanstående rutin beskrivas.

2. Kopiera de viktigaste filerna från BOOT-disketten genom att utföra kommandot **harddoit**.

3. Stoppa systemet (**kill 1**) och starta upp igen manuellt, denna gång från det fasta skivminnet till D-NIX enanvändarmod.

Efter den fullständiga installationen av skivminnet stoppas systemet temporärt med kommandot **kill**.

```
kill 1
```

När texten **System halted** visas på terminalen ska systemet startas från skivminnet. Notera att med de filer som nu finns kan inte systemet starta i fleranvändarmod. Nyckeln skall fortfarande vara i läget OFF (MANUAL) och uppstarten ske i manuell mod upp till enanvändarmod, men nu med winchesterskivminnet som root system genom att enbart RETURN ges som svar på alla frågor vid uppstarten och kommandot **dnix** ges från laddningsprogrammet tills prompten # erhålls.

4. Läs in övriga systemfiler från tar-disketterna med **tar**.

Efter den grundläggande installationen måste övriga filer i systemets grundpaket laddas in från tar-disketterna med resten av systemprogramvaran, vanligen sex disketter. Två disketter innehåller filer och program som är speciella för den aktuella datormodellen, medan fyra disketter innehåller kommandon, som är gemensamma för de olika modellerna.

Exempel:

```
D-NIX 5.2   079-8700-10/ n(7)
Version X.xx
Grundsystem DS90-XX
tar format / 720 kb
DataIndustrier DIAB AB
```

där /n är /2, /3, /4, /5, /6 och /7.

Disketterna läses in i den ordning som anges av numret (/n) med tar-kommandon enligt instruktionerna i medföljande Release Notice för operativsystemet. Normalt ges följande kommando för varje diskett som sättes in.

Exempel:

```
cd /
tar -xvf /dev/mf0
```

5. Systemet kan nu stoppas och startas upp i AUTO-mod och anpassas till användarens miljö genom att definera nya användare och lösenord m m. Skrivarsystemet kan aktiveras.

Efter alla alla disketter är inlästa skall systemet stoppas och åter startas, denna gång i AUTO-mod. Ge först kommandot:

```
/etc/shutdown
```

När texten **System halted** visats på terminalen skall nu nyckeln vridas till läget **AUTO** så att systemet automatiskt startar upp till fleranvändarmod. Om meddelandet 'M680X0 prom' hinner visas på skärmen kan nyckeln behöva vridas först till RESET och därefter direkt till AUTO för att initiera uppstarten.

6. Nu bör ett total backup tas av grundsystemet genom att kopiera hela skivminnet till ett streamer-band.

7. Därefter laddas eventuella optionsprogram in och eventuellt användarens tidigare filer från en backup, varefter en ny totalbackup tas med systemet färdigt för användning.

Exempel på en **hardoit** fil för DS90:

Den långa raden med 'tar' visas nedan uppdelad på grund av begränsad radlängd i manualen. Tecknet 'Ö' (stora Ö) i slutet av raden anger att raden är uppdelad. På bildskärmen kan den se annorlunda ut.

```
# harddoit boot routine

MNT=/mnt
DEV=/dev/si0

dd if=/dev/mf0 of=¤DEV count=1

if Ä `/etc/mount ¤DEV ¤MNT` Å
then
  echo "harddoit: ERROR: Failed to mount ¤DEV"
else

  # Copy everything
  tar cfS - bin boot dev dnix etc tmp usr mf0  Ö
                  .profile ö (cd ¤MNT; tar xf -)
  #
  /etc/umount ¤DEV
fi
```

2.3 Systemavstängning

När systemet stängs av är det viktigt att det görs på ett ordnat sätt. Alla användare måste få möjlighet att logga ut från systemet och alla filer och filsystem måste administreras.

Det finns tre normala sätt att förändra systemets konfiguration:

1. Fullständig avstängning, varvid även spänningen slås av. Detta sker helt automatiskt genom att vrida nyckeln till OFF för alla modeller utom DS90-11. Med DS90-11 måste först systemoperatören ge kommando för avstängning enligt punkt 2 nedan och spänningen slås av manuellt först efter att operativsystemet avslutats fullständigt. Detta sätt kan även användas på övriga datormodeller.

2. Omstart av hela systemet genom kommandot '**/etc/shutdown -k**'. vilket beskrivs i nästa avsnitt. Inom 10 sekunder efter att operativsystemet avslutats, dvs efter att texten ' **OS INFO: System halted'** skrivits ut, kan nyckeln vridas till OFF för att initiera manuell uppstart. I DS90-11 vrids nyckeln till MANUAL.

3. Nedgång från fleranvändarmod till enanvändarmod, vilket görs med kommandot '**/etc/shutdown**'.

Fullständig avstängning av alla modeller utom DS90-11

Detta sköts automatiskt av systemet då nyckeln på frontpanelen vrids från AUTO till OFF då systemet är i en- eller fleranvändarmod.

* Vrid nyckeln från AUTO till OFF. (Ej i DS90-11)

* Nu tar det en stund medan systemet stängs. Filsystemet är avstängt när följande avslutande text visas på huvudkonsolen. Före denna kommer andra systemmeddelanden under avstängningsproceduren.

```
OS INFO:  System halted
```

* 10 sekunder efter att denna text visats stängs automatiskt spänningen av om nyckeln fortfarande är i läget OFF.

* Efter att texten "System halted" visats är det möjligt att vrida nyckeln till RESET och tillbaka till OFF, vilket ger en manuell uppstart. Om nyckeln däremot vrids till AUTO sker en automatisk återstart.

Avstängningen sköts genom att systemet exekverar kommandot **powerfail**. Därvid sker avstängningen automatiskt enligt ungefär samma mönster som med det manuella kommandot **shutdown** enligt beskrivningen i nästa avsnitt.

Fullständig avstängning av DS90-11

Avstängning av DS90-11 måste göras av systemoperatören, inloggad som super-user (root). Följande kommando måste ges från systemets huvudkonsol (/dev/console):

```
/etc/shutdown -k
```

* Nu tar det en stund medan systemet stängs. Filsystemet är avstängt när följande avslutande text visas på huvudkonsolen. Före denna kommer andra systemmeddelanden under avstängningsproceduren.

```
OS INFO:  System halted
```

* Inom 10 sekunder från att denna text visats måste nyckeln ha vridits till läget MANUAL. I DS90-11 kan nyckeln vridas till MANUAL redan innan **shutdown**-kommandot ges. Om nyckeln efter 10 sekunder står i AUTO-läget påbörjas ny automatisk uppstart.

* Därefter (med nyckeln i MANUAL) kan spänningen slås av med huvudbrytaren baktill på systemet.

* Är nyckeln kvar i läget MANUAL sker en manuell uppstart.

Fullständig avstängning från laddningsprogrammet

Avstängning av systemet från bootnivå 0 (boot) och bootnivå 1 (laddningsprogrammet) kan ske direkt genom att stänga av spänningen med huvudbrytaren, under förutsättning att inget program håller på att skriva eller läsa i skivminnet. I dessa lägen kan även systemet omstartas direkt genom att vrida nyckeln till läget RESET och tillbaka till OFF. I början av kapitlet beskrivs hur manuell uppstart sker via bootnivåerna 0, 1, 2 och 3, där 2 och 3 motsvarar en- och fleranvändarnivåerna.

2.3.1 /etc/shutdown

När systemet stängs av är det viktigt att det görs på ett ordnat sätt. Alla användare måste få möjlighet att logga ut från systemet och alla filer och filsystem måste administreras.

Detta sköts med kommandot **/etc/shutdown**. I alla datormodeller utom DS90-11 kan samma funktioner automatiskt utföras med systemprogrammet /etc/powerfail.

Även då systemet enbart skall tas ner till enanvändarmod skall kommandot **/etc/shutdown** användas istället för '**telinit s**', eftersom shutdown även avslutar sådana processer som inte 'telinit' kan avsluta på ett riktigt sätt.

shutdown är ett systemprogram. Dess primära funktion är att avsluta alla processer som körs, på ett ordnat och riktigt sätt. Utan optioner avslutas alla processer och systemet går ner i enanvändarnivå utan att helt stängas. Om optionen '-k' ges till kommandot, kopplas däremot systemet helt ner. En avstängning med shutdown tar normalt mindre än 5 minuter om inga användare finns inloggade.

shutdown går igenom följande steg: Först uppmanas alla inloggade användare att logga ut, via ett utsänt meddelande. De får fem minuter på sig att själva avsluta, annars loggas de ut automatiskt. Alla mountade filsystem uppdateras innan systemet stängs av. Detta måste göras innan systemet laddas igen, för att hålla filsystemet intakt.

Som parameter till /etc/shutdown kan anges en tid i minuter istället för de normala 5 minuter som användarna får för att avsluta. Dessutom kan ett meddelande anges som parameter, vilket visas för användarna tillsammans med upp maningen från shutdown att avsluta.

shutdown kräver superuserprivilegier och att operatören loggat in som 'root' på huvudkonsolen.

Kommandot används på följande sätt för att helt stänga av systemet:

* Skriv **`/etc/shutdown -k`**

* Om inga är inloggade avslutas alla processer direkt på ett ordnat sätt.

* Med inloggade användare får var och en ett meddelande varje minut om att systemet tas ned inom en viss tid. Efter fem minuter tas systemet ned på ett ordnat sätt, varvid alla inloggade automatiskt loggas ut.

Vid fullständing avstängning med '**/etc/shutdown -k**' måste operatören vänta tills systemet skrivit följande på konsolterminalen, innan nyckelbrytaren får vridas till läget OFF/MANUAL och/eller huvudströmbrytaren slås av.

```
OS:INFO: System halted
```

Därefter kan systemet startas upp manuellt via laddningsprogrammet och enanvändarmod upp till fleranvändarmod.

Kommandot används på följande sätt för att gå ner till enanvändarmod:

* Skriv **/etc/shutdown**

* Om inga är inloggade avslutas alla processer direkt på ett ordnat sätt.

* Med inloggade användare får var och en ett meddelande varje minut om att systemet tas ned inom en viss tid. Efter fem minuter tas systemet ned på ett ordnat sätt, varvid alla inloggade automatiskt loggas ut.

Efter att alla processer utom 'init' avslutats går systemet in i enanvändarmod, vilket är systemnivå 's'. Därefter kan enbart huvudkonsolen användas. Denna är alltid samma som den fysiska huvudkonsolen, eftersom /etc/shutdown endast kan startas från den denna.

2.3.2 /etc/powerfail

/etc/powerfail är ett systemprogram, som automatiskt anropas då nyckeln vrids till läget OFF. Programmet stänger av datorn helt tills spänningen automatiskt slås av. Alla processer avslutas och filsystemet uppdateras på samma sätt som med kommandot **/etc/shutdown**, men snabbare och mer direkt. Användare som är inloggade får fortfarande max 5 minuter på sig att logga ut, annars stoppas systemet direkt.

Ovanstående gäller ej DS90-11.

3

3. Program i enanvändarmod, "sas"

3.1 Uppstart av laddningsprogrammet

För att använda de kommandon som beskrivs i detta kapitel måste systemet startas upp på nytt i manuell mod. Vid uppstart läses först ett laddningsprogram in.

Laddningsprogrammet är ett litet operativsystem som enbart används för vissa speciella maskinberoende kommandon.

Även vid automatisk uppstart läses laddningsprogrammet in, men då utförs automatiskt ett kommando som direkt laddar och D-NIX operativsystem från skivminnet. D-NIX startas sedan först upp i enanvändarmod och går sedan automatiskt över i fleranvändarmod.

Vid manuell uppstart passeras följande nivåer. Dessa passeras automatiskt vid automatisk uppstart, upp till den bootnivå som angivit med kommandot /sas/bootpar enligt följande kapitel.

Bootnivå 0: Här väntar det program som ligger i hårdvaran på att operatören skall ange från vilken enhet som laddningprogrammet skall läsas in.

Bootnivå 1: Nu är laddningsprogrammet aktivt och de kommandon som beskrivs i detta kapitel kan användas. Tecknet > skrivs ut som prompt.

Bootnivå 2: Nu har operativsystemet startats. Denna nivå är D-NIX enanvändarnivå.

Bootnivå 3: Detta är D-NIX fleranvändarnivå, vilket är det nivå som automatisk uppstart normalt sker till. Inom denna finns i princip flera systemnivåer, men normalt används bara systemnivå 2.

3.2 /sas/bootpar

Kommando för ändring av startparametrar i NVRAM. Kommandot finns i två versioner. Det ena, **/sas/bootpar**, kan endast användas under laddningssystemet och inte under D-NIX. Därför måste systemet köras upp manuellt till laddningsprogrammet innan **/sas/bootpar** kan användas. **/etc/bootpar** kan användas under D-NIX, men skrivning till NVRAM kan bara göras av en användare med superuser-privilegier.

bootpar är interaktivt. Datorn ställer alltså frågor och operatören svarar. Svar med enbart <RETURN> gör att de gamla värdena som visas inom parentes behålles.

Exempel:

```
> /sas/bootpar

Console baudrate (13)       Anger baudrate vid manuell
                            uppstart i boot-läge och i
                            laddningsprogrammet.
                            Värde: 13 => 9600 Baud, vil-
                            ket är standard.

Boot level       (3)        AUTO start-nivå 3 (fleranvän-
                            darmod)

Boot device      (5,16)     5=si=winchester.
                            16=intern winchester.

DS90        10/10E      11        20/21

            (5,16)      (5,0)     (5,0)

Root device      (255,255) Parametern används ej.

Swap device      (255,255) Parametern används ej.

Pipe device      (255,255) Parametern används ej.

Mirror device    (255,255) Parametern används ej.
```

Time zone	(60)	I minuter relativt GMT (60 = svensk normaltid, 120 = svensk sommartid) Används ej i D-NIX då parametern TZ satts från filen /etc/timezone. Däremot används denna parameter om ett kommando direkt exekveras istället för en kommandoavkodare enligt filen /etc/passwd.
US daylight s.t	(N)	N=Nej. Kan ej användas i Sverige. Sommartid anges istället genom att ändra i filen /etc/timezone.
CPU clock freq.	(16670000)	Ges i Hz, 16.67 MHz i DS90-20.

DS90	10/10E	11	20/21
	10000000	10000000	16670000

Satisfied	Svara **y(es)** om rätt.
Write to NVRAM	Svara **y(es)** om data skall sparas.

3.2 Förebyggande underhåll

Detta kommando ska endast användas av systemadministratören och endast på skivminnen som inte är "mountade" i systemet. Kommandot skall användas under laddningsprogrammet, i bootnivå 1 efter manuell uppstart, om wichesterskivminnet skall testas.

Skivminne, både disketter och winchesterminne, kan testas med kommandot **fsck**. Normalt förstörs ingen data.

Om systemet upptäcker fel i filstrukturen, rapporteras detta. Med kommandot **fsck** är det möjligt att försöka reparera den filstruktur som är förstörd.

Test av root-filsystemet från stand-alone-mod.

```
>/etc/fsck -rr si(X,Y)

 Checking /dev/si0
 File system:              Volume:
 Volume size =   82Mb,
 Block size = 2048,13312 I-nodes (13312 cont.)

 Phase 1 - Check I-nodes
 Phase 2 - Check Pathnames
 Phase 3 - Check Unreferenced files
 Phase 4 - Check Link Counts
 Phase 5 - Check Bitmap

 Total number of files:    662
 Total number of blocks: 41472,  37136 - free blocks
```

/etc/fsck	Kommandot självt.
-rr	Ska alltid anges i stand-alone-mod.
si(X,Y)	Den enhet som filsystemet befinner sig på. Denna ska anges i samma form som vid manuell uppstart från enheten.
Övrig text	Skrivs ut av systemet som en upplysning om vad som kontrolleras.

Följande värden gäller för **X,Y** för de olika modellerna:

DS90 -	10/10E	11	20/21
X,Y	(16,0)	(0,0)	(16,0)

Om kommandot **fsck** hittar filer eller bibliotek som inte är knutna till filsystemet, länkas dessa vid reparationen till biblioteket /lost+found. I detta bibliotek finns det alltså möjligheter att återfinna data som gått förlorad vid reparationen.

Ett inte mountat filsystem, t ex på en diskett, kan testas från D-NIX, men då ska istället enhetsnamnet anges som argument. För test av ett D-NIX filsystem på diskett ges följande kommando:

```
/etc/fsck /dev/mf0
```

Se i övrigt beskrivning av **fsck** i **AH**.

4

4. Filer och hjälpprogram för systemunderhåll

BootPROM
/etc/bcheckrc
/dev/bclock
/etc/bootpar
/etc/brc
/dev/console
/bin/date -B
/dev
/etc/dialups
/etc/d_passwd
/etc/getty
/etc/gettydefs
/etc/group
/etc/init
/etc/inittab
/etc/ioctl.syscon
/etc/issue
/bin/login
/bin/lp
/bin/lpr
.mail.rc
/usr/lib/mail/mail.rc
/etc/mknod
/etc/mkuser
/etc/motd
/dev/nvram
/bin/passwd
/etc/passwd
/etc/powerfail
/etc/profile
.profile
/etc/rc
/etc/rmuser
/etc/setspeed
/etc/shutdown
/bin/stty
/dev/syscon
/dev/systty
/bin/telinit
/etc/termcap
/usr/lib/terminfo
/etc/timezone

4. Filer och hjälpprogram för systemunderhåll

Allmänt

Nedan beskrivs enheter och filer som innehåller parametrar eller kommandon för systemstart eller inloggning och som den systemansvarige kan vilja ändra på. Nyttiga hjälpprogram (med fetstil nedan) beskrivs kortfattat.

BootPROM	Litet laddningsprogram i en fast minneskrets. Kan ej ändras.
/etc/bcheckrc	Kommandofil som bearbetas vid systemstart.
/dev/bclock	Batteriuppbackad klock-krets.
/etc/bootpar	Kommando för ändring av /dev/nvram.
/etc/brc	Kommandofil som bearbetas vid systemstart och rensar mount-tabeller.
/dev/console	Systemets huvudkonsol. Skall alltid vara samma som /dev/systty.
/bin/date -B	Kommando för att ställa batteriklockan.
/dev	Bibliotek med fysiska enheter.
/etc/dialups	Lista över terminalportar där sekundärt lösenord krävs.
/etc/d_passwd	Lista med sekundära lösenord för inloggning via de terminalportar som listats i /etc/dialups.
/etc/getty	Kommando som sätter upp kommunikationsparametrar och startar inlogging. Exekveras automatiskt via /etc/inittab.
/etc/gettydefs	Tabell med baudrate och andra kommunikationsparametrar för terminaler vid inlogging.
/etc/group	Lista över användargrupper med grupplösenord m m.

/etc/init	Systemprocess som är urfadern till alla andra i operativsystemet.
/etc/inittab	Tabell med kommandon som ska göras vid systemstart och vid övergång mellan systemnivåer, t ex mellan enanvändarmod och fleranvändarmod.
/etc/ioctl.syscon	Datafil med temporära terminalparametrar som används i enanvändarmod.
/etc/issue	Fil med systemmeddelande som visas FÖRE inloggning, före login-frågan.
/bin/login	Kommando för inloggning, som automatiskt exekveras av /etc/getty.
/bin/lp	Program för att ta emot utskriftsbegäran som skall köas och skrivas ut med spoolersystemet **lp**. (Standard)
/bin/lpr	Program för att ta emot utskriftsbegäran som skall köas och skrivas ut med spoolersystemet **lpr**.
.mail.rc	Fil med användarens egna alias-namn för kommandot **mail**.
/usr/lib/mail/mail.rc	Fil med globala alias-namn för kommandot **mail**.
/etc/mknod	Kommando för att lägga till nya fysiska enheter.
/etc/mkuser	Kommando för att lägga till nya användare.
/etc/motd	Aktuellt systemmeddelande. Visas EFTER inloggning.
/dev/nvram	Ändringsbara startparametrar i minneskrets.
/bin/passwd	Kommando för ändring av lösenord.
/etc/passwd	Lista över användare med lösenord m m.
/etc/powerfail	Kommandofil som bearbetas då nyckelbrytaren på frontpanelen vrids till OFF, varefter spänningen automatiskt slås av.

/etc/profile	Gemensam kommandofil som automatiskt exekveras före .profile efter inloggning.
.profile	Användarens kommandofil som exekveras vid inloggning.
/etc/rc	Kommandofil som bearbetas vid övergång till fleranvändarmod.
/etc/rmuser	Kommando för att ta bort en användare.
/etc/setspeed	Kommando för att ändra överföringshastighet och andra kommunikationsparametrar för en icke aktiv port.
/etc/shutdown	Kommando som kan ges av operatören för att stänga av systemet eller övergå till enanvändarmod.
/bin/stty	Program för temporär ändring av kommunikationsparametrar efter inlogging.
/dev/syscon	Den virtuella systemkonsolen, dvs den som gäller i enanvändarmod.
/dev/systty	Systemets huvudkonsol. Skall alltid vara samma som /dev/console.
/bin/telinit	Kommando för att byta systemnivå enligt filen /etc/inittab.
/etc/termcap	Parameterfil med terminalstyrtecken, kompatibelt med tidigare versioner.
/usr/lib/terminfo	Bibliotek med parameterfiler, som innehåller styrtecken för olika terminaltyper enligt shellvariabeln TERM. Dessa filer är kompatibla med filen /etc/termcap vid leverans av systemet.
/etc/timezone	Sträng med tidsskillnad för definition av lokal tid.

OBS! att samtliga kommandon skall ges med gemena (små) bokstäver!

För mer information om kommandon, se **AH**.

BootPROM

Ett litet program som laddar in ett startprogram. Startprogrammet kan antingen vara på den enhet "Boot device" som anges i NVRAM eller, i manuell mod, på den enhet som operatören anger från tangentbordet. Programmet BootPROM ligger i en fast minneskrets i systemet och kopieras över till datorns arbetsminne innan första systemstart. Här ligger även vissa testrutiner för hårdvaran som utförs innan startprogrammet laddas.

/etc/bcheckrc

Denna kommandofil exekveras automatiskt av systemet endast första gången övergång sker till fleranvändarmod efter spänningstillslag och inladdning, styrt av /etc/inittab.

I standard UNIX ligger här kommandon för att testa filsystemet och för att sätta systemklockan. Detta system testar automatiskt anslutet filsystem och sätter systemklockan från batteriklockan vid uppstart, varför detta inte behöver göras här.

Example of /etc/bcheckrc:

```
# ***** This file has those commands necessary to
# check the file system, date, and anything else that
# should be done before mounting the file systems.
#
# Warning: Since the root filesystem may be corrupted,
# do not create files or use here documents, etc. (ie.
# `date`) until check is complete!
#
# Since the system has automatic file checking and date
# setting this file is not used.
#
```

/dev/bclock

Batteriuppbackad klockkrets, som sitter på CPU-kortet. Vid systemstart laddas systemets klocka automatiskt med datum och tid från batteriklockan. Batteriklockan kan läsas med **cat** /dev/bclock och innehåller datum och tid i GMT. För ändring av batteriklockan används kommandot **date -B**.

/etc/bootpar eller /sas/bootpar

Kommando för ändring av startparametrar i /dev/nvram. **/etc/bootpar** kan användas under D-NIX, men skrivning till NVRAM kan bara göras av en användare med superuser-privilegier. Normalt bör emellertid ändring av dessa parametrar göras i stand-alone-mod med **/sas/bootpar**.

bootpar är interaktivt. Datorn ställer alltså frågor och operatören svarar. Svar med enbart <RETURN> gör att de gamla värdena som visas inom parentes behålles.

Se **/sas/bootpar** i kapitel 3 för detaljer och exempel!

/etc/brc

Denna kommandofil exekveras automatiskt då systemet startas efter spänningstillslag och inladdning, styrt av /etc/inittab. /etc/brc rensar mount-tabellerna i systemet.

Exempel på /etc/brc :

```
# ***** This command file's function is to remove the
# ***** file /etc/mnttab
#
if Ä -r /etc/mtab Å
then
  rm -f /etc/mtab
fi
if Ä -r /etc/mnttab Å
then
  rm -f /etc/mnttab
fi
```

/dev/console och /dev/systty

Dessa enhetsnamn är synonyma, skall alltid vara länkade till samma fysiska serieport och anger var systemets huvudkonsol finns. Jämför även /dev/syscon, som är virtuell systemkonsol i enanvändarmod.

/bin/date -B

Kommando för ändring av batteriklockan. Systemklockan ändras samtidigt. systemklockan sätts automatiskt vid systemstart, men kan även ändras med kommandot **date** utan optionen -B.

OBS! **date -B** ges alltid med GMT-tid, medan **date** (utan -B) alltid arbetar med lokaltid.

Om batteriklockan går för fort eller sakta justeras systemklockan automatiskt med en justeringskonstant. Konstanten beräknas på nytt varje gång systemklockan sätts med **date** och nollställs då batteriklockan ställs med **date -B**. Därför skall batteriklockan bara ställas en gång och därefter justeras eventuell avvikelse med **date** utan optionen -B. Ju längre tid det går mellan första inställningen (med -B) och sista (utan -B) desto bättre blir justeringskonstanten. Se i övrigt kommandot **date** i **AH**.

Omställning för **sommartid** måste göras i filen /etc/timezone, som laddas till shellparametern TZ vid inloggning. Se beskrivningen längre fram i kapitlet.

Exempel 1: Ändring av batteriklockan.

Nytt värde 1986-09-24 10.30.25 ges med GMT-tid som:

```
date -B 8609241030.25
```

Exempel 2: Ändring av systemklockan, varvid ny justeringskonstant beräknas, utgående från sista gången batteriklockan ställdes.

Nytt värde 1986-09-24 11.30.25 ges med lokal tid som:

```
date 8609241130.25
```

/dev

Detta bibliotek innehåller inte vanliga filer utan används för åtkomst till fysiska enheter i systemet. Se kapitel 15.

/etc/dialups

Lista över terminalportar där extra lösenord skall krävas (dialup password). Vid inloggning till någon av dessa terminalportar kräver systemet, förutom det normala lösenordet, att användaren skall ange ett extra lösenord som skall finnas definierat i filen /etc/d_passwd. Finns inte användaren definierad i /etc/d_passwd avbryts inloggningen. Se i övrigt beskrivningen av **login** i **AH**.

Exempel på /etc/dialups fil:

```
/dev/tty03
/dev/pk00
/dev/modem1
```

/etc/d_passwd (OBS!!! Understrykningstecken mellan d och passwd)

Lista över användare som får logga in via de terminalportar som listats i filen /etc/dialups. Här anges det extra lösenord som skall användas vid inloggning via dessa terminalportar.

/etc/d_passwd är en fil med samma format på fälten som filen /etc/-passwd. Endast de fyra första fälten krävs, inklusive avslutande kolon. Användarnamnet måste stämma med de numeriska ID-numren enligt filen /etc/passwd.

Kommandot **passwd** med -f optionen används för att ändra lösenordet i /etc/d_passwd.

OBS!! Inga åldringskoder får finnas i fältet med lösenordet. Jämför /etc/passwd.

Exempel på /etc/d_passwd fil:

```
jag:5Xu5K8hdfK7as:110:110:Sven Svensson
kalle:kJie3dKd7csHk:111:110:Kalle Karlsson
```

/etc/getty

Detta kommando startas automatiskt av 'init', enligt raderna i /etc/-inittab, för att initiera anslutna terminaler och starta inloggning. Se /etc/inittab för exempel.

Baudrate och andra kommunikationsparametrarna som skall användas tas från tabellerna i /etc/gettydefs, där även en textsträng anges, som visas som inloggningsfråga. Där kan även ytterligare textsträngar anges, avskilda med kommatecken. Dessa är då frågetexter som kommandot **login** kommer att använda efter att /etc/getty startat login-processen.

/etc/getty kan prova olika kommunikationsparametrar genom att tabellerna i /etc/gettydefs kan länkas ihop. Byte till nästa tabell sker då användaren försöker mata in sitt namn, om systemet upptäcker överföringsfel eller ett break-tecken.

/etc/getty startas normalt upp via kommandot **nice** för att vanliga terminaler skall ha lägre prioritet än huvudterminalen. Huvudterminalen bör ha högre prioritet, för att underlätta för systemoperatören vid systemunderhåll eller felsökning.

Efter utloggning återstartas /etc/getty av 'init' igen för ny inloggning.

Kommandosträngen för att starta /etc/getty är:

```
/etc/getty (-h) (-t timeout) line (speed ( type ( ld )))
```

eller för testning av en fil formatterad som /etc/gettydefs:

```
/etc/getty -c filnamn
```

där:

`-h` anger att /etc/getty inte ska göra 'hangup' före aktiveringen av linjen. Hangup innebär att utsignalen DTR i seriella kontakten släpps för att eventuella modem säkert skall kopplas ner från föregående inloggning.

`-t` anger att /etc/getty ska vänta maximalt 'timeout' sekunder på en inlogging. Därefter ska linjen kopplas ner. Detta säkerställer nedkoppling av modem linjer om någon ringer in, men inte loggar in.

`line` Detta är den fysiska enhet dit terminalen är kopplad. Här anges en enhet i biblioteket /dev, t ex tty02.

speed Detta är en pekare som visar på en "rad" i filen /etc/gettydefs. Där anges vilka terminalparametrar som ska provas först och hur loginfrågan ska se ut och vilka parametrar som ska provas sedan om det behövs. Om denna parameter utelämnas, används den första raden i filen /etc/gettydefs. Första raden används även om ingen rad motsvarande 'speed' finns i /etc/gettydefs. Om filen /etc/gettydefs saknas, används 300 baud.

type Används ej. I vissa system anges här om speciella terminaldrivrutiner skall användas.

ld Används ej. I vissa system anges här om speciella kommunikationsprocedurer skall användas.

-c För att testa formatet på filen /etc/gettydefs, eller en ny ändrad fil som ska användas, kan /etc/getty ges från tangentbordet som ett kommando med flaggan -c. Då testas tabellerna och rapporterar eventuella fel. Med korrekta tabeller skrivs parametrarna ut.

Inloggningen sker i följande ordning när /etc/getty startas:

* Terminalparametrarna sätts enligt /etc/gettydefs.

* En system-identifikation skrivs ut. Denna är nodnamnet som kan ses med kommandot 'uname -n' och skapas med /etc/mkcfig.

* Texten i filen /etc/issue skrivs ut.

* Inloggningsfrågan i /etc/gettydefs visas.

* Användaren matar in sitt användarnamn på terminalen. Under tiden kan /etc/getty ställa om vissa terminalparametrar.

* Efter att användarnamnet matats in, men innan '**login**' anropas, ställer /etc/getty om terminalparametrarna enligt den andra parametergruppen för 'raden' i /etc/-gettydefs.

* Processen '**login**' startas med användarnamnet som parameter.

* '**login**' läser och testar inloggningskoden (password) samt sätter upp vissa shellvariabler innan den shell eller annan process startas som angivits i /etc/passwd.

Övriga parametrar som kan ändras av /etc/getty beror av de tecken som mottages vid inmatning av användarnamnet och är följande:

Om namnet avslutas med koden RETURN (0D Hex) istället för det normala New-Line (0A Hex) sätts terminalparametrarna så att RETURN alltid konverteras till New-Line vid inläsning från tangentbordet.

Om namnet inte innehåller någon liten bokstav, sätts parametrarna så att alla stora bokstäver i fortsättningen översätts till små vid inläsning från terminalen.

Om användaren använder CTRL-H (08 Hex) som raderingstecken eller CTRL-U (15 Hex) som tecken för att sudda raden, sätts parametrar upp så att dessa kan användas som ERASE och/eller KILL-tecken vid inmatning. Standard är annars att ERASE är # och KILL är É. I vilket fall som helst ändras dessa efter inloggningen till CTRL-H respektive CTRL-X av ett **stty**-kommando som normalt finns i filen .profile.

Om /etc/getty upptäcker någon av tecknen '_', '¤', '&', '/, eller '!' sätts parametrar upp för att använda dem enligt standarden för ESS2 protokoll. '_' = erase, '&' = abort, '¤' = kill, '/' or '!' = line terminator.

/etc/gettydefs

Denna fil innehåller tabeller för inloggning med kommandot /etc/getty, för att definiera terminalparametrar. En parameter till kommandot /etc/-getty är 'speed', vilken används som pekare till en "tabellrad" i filen /etc/gettydefs.

Varje tabellrad följs av en tom rad i filen. En "tabellrad" kan sträcka sig över flera textrader.

Om /etc/getty startas utan parametern 'speed' eller om 'speed' ej återfinns i filen, används första tabellraden i /etc/gettydefs.

De olika fälten är begränsade av tecknet '#' och är:

speed # paraml # param2 #Login prompt # next

eller

speed # paraml # param2 #Login,Lopt,Popt,Dopt # next

speed Motsvarar parametern 'speed' i /etc/getty.

paraml Terminalparametrar i ett format som liknar parametrarna till kommandot stty, men med stora bokstäver och alltid bokstaven B före baudrate-talet. /etc/getty sätter upp dessa innan systemmeddelanden visas och användarnamnet läses in. Dessa parametrar används tills **login**-kommandot ska startas. För parametrarna, se kommandot stty i AH.

param2 Dessa terminalparametrar sätts upp efter inmatning av namnet, men innan loginkommandot startas.

Login,Lopt,Popt,Dopt
Detta fält kan innehålla upp till fyra olika frågesträngar, separerade av kommatecken. Härigenom kan valfritt språk användas vid inloggningsrutinerna. Notera att eventuella mellanslag i dessa strängar visas som de är i detta fält. Den första strängen används av /etc/getty för första frågan efter användarens namn. De tre följande strängarna sänds som optionerna -L, -P och -D, i denna ordning, till kommandot **login** tillsammans med det inlästa användarnamnet. **Login** använder därefter dessa frågesträngar för att fråga efter lösenord och eventuellt nytt användarnamn. Se kommandot **login** för detaljer.

next Detta är en pekare till 'speed' i en annan tabell i /etc/- gettydefs och är första parametern i nästa tabell som /etc/getty ska prova vid inmatningsfel.

Exempel på tabeller i filen /etc/gettydefs:

På grund av begränsad radlängd i manualen delas långa rader i listningen nedan. På bildskärmen kan de se annorlunda ut.

```
console# B9600 HUPCL PARENB CS7 OPOST ONLCR # B9600
        SANE IXANY TAB3 ECHOE #Console Login: #console

1200# B1200 HUPCL PARENB CS7 # B1200 SANE IXANY TAB3
         ECHOE #login: #300

300# B300 HUPCL PARENB CS7 # B300 SANE IXANY TAB3
        ECHOE  #login: #1200

2400# B2400 # B2400 SANE IXANY TAB3 ECHOE #login: #300

4800# B4800 HUPCL PARENB CS7 # B4800 HUPCL SANE IXANY
       TAB3 #login: #9600

9600# B9600 HUPCL PARENB CS7 # B9600 SANE IXANY TAB3
         ECHOE #login: #9600

19200# B19200 HUPCL PARENB CS7 # B19200 HUPCL SANE
         IXANY TAB3 ECHOE #login: #9600
```

Exempel på en rad i /etc/gettydefs med frågesträngar för **login**-kommandot på svenska. Notera att mellanslag kan ingå i dessa strängar, lämpligt särskilt som sista tecken för att separera frågan från användarens svar.

```
sw96# B9600 HUPCL PARENB CS7 # B9600 SANE IXANY
   TAB3 ECHOE #Namn: ,Namn: ,Lösen: ,Modemkod: #sw96
```

/etc/group

Lista över användargrupper. Nedan ges ett exempel där de olika fälten förklaras.

Exempel:

```
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
mail::6:root
rje::8:rje,shqer
daemon::12:root,daemon
vi::110:jag,kalle
```

Filen innehåller en rad per grupp. Raden har fyra fält, separerade med ':'. Det sista fältet är endast en kommentar.

Fält 1:
Gruppnamn. Detta används i kommandot **newgrp** när en användare vill byta grupp efter inloggningen. Gruppen 'other' med numret 1 är en standard-grupp till vilken alla användare hör, om ingen grupp anges då användaren skapas med kommandot **/etc/mkuser**.

Fält 2:
Krypterad version av lösenordet för gruppen. Lösenordet används endast i kommandot **newgrp** när en användare byter grupp efter inloggningen. Ett lösenord för en användargrupp kan endast definieras med kommandot **/etc/mkuser**.

Fält 3:
Gruppnummer.

Fält 4:
Kommentar, där den systemansvarige lämpligen listar de användare som ingår i gruppen. **/etc/mkuser** skriver automatiskt in användarnamnen då de skapas. Namnen skiljs åt med kommatecken.

/etc/init

Processen '**init**' är den första process som laddas in och startas då operativsystemet startas upp. 'init' stannar sedan kvar i systemet som urfadern till alla andra processer. Dess huvudsakliga funktion är att starta upp andra processer enligt informationen i systemfilen **/etc/inittab**. T ex startar den upp kommandot /etc/getty mot varje terminalanslutning för inlogging av användare i fleranvändarmod. Den kan även starta initieringsprocesser såväl som allmänna systemprocesser som ska arbetar i bakgrunden, t ex nätverksprogram eller speciella filhanterare. 'init' kan återstarta processer, som t ex inloggning efter att en användare loggat ut ur systemet.

I filen /etc/inittab anges ett kommando på varje rad och varje rad innehåller en siffra (i 2:a fältet) som anger på vilken systemnivå som detta kommando skall aktiveras. Flera systemnivåer kan anges på varje rad, varigenom olika systemkonfigurationer kan skapas, beroende på vald systemnivå (="run-level" på engelska). Systemnivå 2 är standard för fleranvändarmod. Möjliga systemnivåer är 0 .. 6 samt S, där S (eller s) är enanvändarmod. För att byta systemnivå kan kommandot **telinit** användas. Nivå 0 är reserverad och inga processer får definieras på nivå 0 i /etc/inittab. Nivå 0 används nämligen av **/etc/shutdown** medan olika optionsprogram stängs av på ett kontrollerat sätt.

Det 3:e fältet på varje rad markerar ett villkor för hur processen ska arbeta. Detta och övriga fält på raderna beskrivs närmare nedan för /etc/inittab.

Vid manuell uppstart av systemet upp till enanvändarmod, läses inte filen /etc/inittab utan systemet startar direkt i enanvändarmod utan den fråga som beskrivs nedan. På bootnivå och i laddningsprogrammet används alltid den fysiska huvudkonsolen /dev/console och standardparametrarna 9600 Baud, 7 databitar, jämn paritet oberoende av konfigurationsfilerna i systemet. I enanvändarmod används däremot /dev/syscon uppsatt enligt filen /etc/ioctl.syscon.

Vid automatisk systemstart upp till fleranvändarmod startas 'init' direkt på den systemnivå som markerats med villkoret 'initdefault' i /etc/inittab. Om inget 'initdefault'-villkor finns eller om ingen /etc/inittab finns. skriver 'init' ut en fråga på den virtuella systemkonsolen (/dev/syscon) och ber operatören ange en systemnivå i form av ett tecken.

```
INIT: ENTER RUN-LEVEL(0-6,s or S): _
```

Detta kan vara en siffra i området 0..6 eller bokstaven 's' (eller 'S'). Systemnivå 2 är den normala fleranvändarnivån, medan bokstaven 's' anger enanvändarmod ("single-user-mode").

/dev/syscon kan vid denna fråga vara länkad till en annan terminal än systemets huvudkonsol. Ett meddelande ges då till systemets huvudkonsol /dev/systty (=/dev/console) var den virtuella systemkonsolen är länkad.

Om operatören sitter vid systemets fysiska huvudkonsol och frågan efter systemnivå inte kommer, kan det bero på att /dev/syscon är länkad till en annan terminal. Operatören kan då skriva tecknet 'DEL' på systemets huvudkonsol, varvid 'init' automatiskt länkar om /dev/syscon dit i samband med frågan om systemnivå.

Övergång från enanvändarmod till fleranvändarmod görs antingen genom kommandot **telinit 2** eller genom att avsluta med CTRL-D (End-of-file). Notera att CTRL-D kan inte användas om systemet startats upp manuellt och aldrig varit uppe i fleranvändarmod. Vid avslutning med CTRL-D kommer 'init' att ge en ny fråga enligt ovan efter systemnivå, varvid **2** anges för att gå till normal fleranvändarmod.

Endast första gången systemet övergår till fleranvändarmod efter uppstart, startas de processer som i /etc/inittab markerats med villkoret **'boot'** eller **'bootwait'** (under förutsättning att systemnivån som angivits för dem är rätt). Dessa startas alltid före andra processer oberoende av positionen i filen /etc/inittab.

Efter att 'init' startat alla processer enligt /etc/inittab väntar 'init' på en av följande händelser:

* Någon process som skall återstartas dör. T ex om en användare loggar ut, ska 'init' åter starta inlogging på den berörda terminalen.

* En signal att systemet ska stängas av, dvs. en 'powerfail'-signal (SIGPWR). Då exekverar 'init' de processer som i /etc/inittab markerats med villkoret 'powerfail', fortfarande beroende på systemnivån som markerats på raden i /etc/inittab.

* En signal kommer från t ex kommandot **'telinit'** med en ny systemnivå (eller samma). Då går 'init' åter genom /etc/inittab och vidtar åtgärder enligt den givna systemnivån.

Om någon tidigare startad process, t ex inloggning på en terminal, skall avslutas vid en ändring av systemnivån, sänder 'init' först en varning (SIGTERM) till processen och efter 20 sekunder avslutas processen med signalen SIGKILL.

Notera att endast processer som startats av 'init' kan avslutas. Om någon process startat upp andra processer i bakgrunden och sedan själv avslutas, känner 'init' inte till dessa och kan inte avsluta dem. Jämför kommandot **/etc/shutdown** och filen /etc/rc.

/etc/inittab

Denna fil innehåller information om alla processer som skall startas upp automatiskt av systemet av 'init'-processen vid övergång till fleranvändarmod eller vid val av en annan systemnivå inom fleranvändarmoden.

Varje process som ska startas beskrivs med fyra fält på en "rad". Liksom i shell kan en "rad" vara längre än 80 tecken genom att tecknet Ö i slutet av textraden anger att informationen fortsätter på nästa textrad. Max 512 tecken tillåts i en processbeskrivning (en "rad").

Om /etc/inittab ändras kan systemet tvingas läsa filen igen genom kommandot '**telinit q**'.

De olika fälten är begränsade av tecknet ':' och är:

```
id:rstate:action:process # kommentar
```

Fältet **id:**
ID-kod för raden. Upp till fyra tecken. Alla rader måste ha olika ID-koder. De används intern i 'init'.

Fältet **rstate:**
En lista över de systemnivåer där denna process ska vara aktiv. 'rstate' kan vara en eller flera siffror 0..6 som representerar systemnivåer i fleranvändarmod. Om fältet 'rstate' är tomt betyder det att processen är aktiv på alla nivåer. Alternativt kan 'rstate' anges som a, b och/eller c. Processer som markerats så startas enbart då motsvarande nivå-kod ges via kommandot 'telinit', utan att den verkliga systemnivån ändras. Nivå 0 får ej anges, då den i D-NIX representerar en nivå där inga processer i /etc/inittab är aktiva.

Fältet **action:**
Villkor som anger hur processen ska behandlas av 'init'. Villkoren anges som en textsträng. Möjliga villkor är enligt nedan. Om inget annan sägs stoppas respektive process om systemnivån ändras så att 'rstate' ej längre överensstämmer med systemnivån.

boot Endast starta processen vid första uppstart till fleranvändarmod enligt givna nivån. 'rstate' bör vara tomt.

bootwait Samma som 'boot' men vänta tills processen är klar innan nästa process startas enligt /etc/inittab.

initdefault Detta är enbart en markering som visar att den systemnivå som givits i motsvarande 'rstate'-fält skall gälla vid automatisk systemstart till fleranvändarmod. Om ingen rad med 'initdefault' finns frågar 'init' efter systemnivå enligt beskrivningen för /etc/init ovan.

sysinit Innan 'init skriver eller läser på systemkonsolen, t ex för en fråga om ny systemnivå, exekveras processer som markerats med 'sysinit', om nivån i 'rstate' är rätt. 'init' väntar alltid tills denna process är klar.

off Motsvarande process ska ej vara aktiv. Startas ej och stoppas om den redan är aktiv.

once Starta processen då systemnivån blir den nivå som givits i fältet 'rstate'. Om processen redan är igång görs ingenting om systemnivån är rätt.

wait Samma som 'once' men vänta tills processen är klar innan nästa process startas enligt /etc/inittab.

respawn Detta villkor används normalt för att starta inloggning på terminalerna i systemet och innebär att processen (vanligen /etc/getty) startas utan att 'init' väntar på att avslutas. Dessutom återstartas processen efter att den avslutats, styrt av nivålistan i 'rstate'. I filerna /etc/utmp och /etc/wtmp sparar 'init' information om de processer som startats och avslutats.

ondemand Samma som 'respawn' men används bara med nivå a, b eller c för aktivering med kommandot 'telinit'. Dessa processer stoppas ej automatiskt av 'init' vid byte av systemnivå. De stoppas bara om motsvarande "rad" i /etc/inittab markeras med 'off' eller tas bort, eller om systemet går ner i enanvändarmod.

powerfail Denna process exekveras då signalen SIGPWR indikerar att spänningen ska stängas av. Denna signal ges bl a då nyckelbrytaren på frontpanelen vrids till läget OFF. Denna funktion används ej i DS90-11.

powerwait Samma som 'powerfail', men vänta tills processen är klar innan nästa process kan startas enligt /etc/inittab.

Fältet **process:**
Detta är det kommando som ska exekveras och kan skrivas i standard shell-format. Vid exekveringen skapas ett kommando från strängen 'process' med följande format:

```
sh -c 'exec process'
```

Strängen 'process' kan innehålla optioner och argument till kommandon.

Fältet **kommentar:**
Kommentarer kan läggas in i slutet av varje "rad", efter tecknet # enligt normal shell-standard. Kommentarer som avslutar rader med **/etc/getty**-kommandon, visas med kommandot **who** som förklarande text tillsammans med användarnamnet, och kan till exempel innehålla upplysningar om var respektive terminal är placerad.

Notera att bootnivå 3 motsvarar den standardiserade systemnivån 2 i /etc/inittab, dvs fleranvändarnivån och att filen /etc/inittab inte alls används vid uppstart direkt till enanvändarmod (bootnivå 2).

Systemnivå 0..6 är alla på systemets fleranvändarnivå. I systemet är nivå 0 reserverad och skall vara utan aktiva processer. Några systemnivåer kan som i exemplet konfigureras så att endast huvudkonsolen är aktiv, där den systemansvarige kan ha ensam tillgång till systemet. Denna nivå kan väljas med kommandot **telinit 3**, med /etc/inittab enligt exemplet.

Notera att alla vanliga terminaler normalt startas med lägre prioritet än huvudkonsolen (kommandot nice -16).

Exempel på tabeller i /etc/inittab:

På grund av begränsad radlängd i manualen delas långa rader i listningen nedan. På bildskärmen kan de se annorlunda ut.

```
is:2:initdefault:
bl::bootwait:/etc/bcheckrc </dev/console
                          >/dev/console 2>&1 ; #bootlog
bc::bootwait:/etc/brc 1>/dev/console 2>&1 #bootrun
sl::wait:(rm -f /dev/syscon;ln /dev/systty
                     /dev/syscon;) 1>/dev/console 2>&1
rc:2:wait:/etc/rc 1>/dev/console 2>&1 ; #run com
err::boot:/usr/lib/errdemon -c /usr/adm/errfile
                           /usr/adm/errmessages
pf::powerfail:/etc/powerfail </dev/console 1>&0 2>&0
net:2:off:nice -16 /usr/lib/net/netman -F
ra:2:off:nice -16 /usr/lib/net/raccess -F
db3:2:off:nice -16 /usr/bin/dblm /mimerdbl
                           /usr/mimer/sysdb/sysdb
co:123456:respawn:/etc/getty console console
t2:2:respawn:nice -16 /etc/getty tty02 9600
t3:2:respawn:nice -16 /etc/getty tty03 9600
t4:2:off:nice -16 /etc/getty tty04 9600
t5:2:off:nice -16 /etc/getty tty05 9600
t6:2:off:nice -16 /etc/getty tty06 9600
t7:2:off:nice -16 /etc/getty tty07 9600
t8:2:off:nice -16 /etc/getty tty08 9600
t9:2:off:nice -16 /etc/getty tty09 9600
t10:2:off:nice -16 /etc/getty tty10 9600
t11:2:off:nice -16 /etc/getty tty11 9600
```

/etc/ioctl.syscon

Denna fil innehåller de terminalparametrar som automatiskt sätts för den virtuella systemkonsolen (/dev/syscon) i enanvändarmod. I fleranvändarmod är systemkonsolen alltid /dev/console, eller med ett annan namn /dev/systty.

Filen skapas vid övergång från fleranvändarmod till enanvändarmod enligt de terminalparametrar som just då är inställda. Parametrarna anges i en form som direkt kan användas som kommandoparametrar till kommandot **stty** och består av en rad med parametrar i hexadecimal form separerade med kolon ':'.

/etc/issue

Detta är en textfil med lämplig systemidentifikation eller systemmeddelande, som automatiskt visas av /etc/getty före inloggning, men efter raden med systemets nodnamn.

Efter denna text visas den inloggningsfråga, som definierats i /etc/gettydefs, innan användaren börjar mata in sitt användarnamn.

/bin/login

Detta är kommandot som sköter själva inloggingen, med test av lösenordet (password) och uppsättning av de shellvariabler som är standard (t ex HOME, PATH m fl.).

Kommandot anropas av /etc/getty men kan även användas direkt i formen '**exec login**' för att tvinga fram en utloggning och direkt återinloggning som en annan användare utan att terminalparametrarna ställs om med /etc/getty. Denna form kan enbart användas från den primära shell som startades vid inloggning och inte från shell som startats med kommandot **sh**.

/bin/lp

Program för att ta emot utskriftsbegäran till kön i skrivarsystemet (**lp**-spooler-systemet). I **lp**-kommandot anges med en option på vilken logisk skrivare som utskriften skall skrivas ut. Alternativt kan en skrivargrupp anges, varvid utskriften sker på första lediga skrivare i gruppen. Det som ska skrivas kan även riktas till **lp**-kommandot med en pipe, varefter det läggs i skrivarkön.

Utskrifter i skrivarkön tas om hand av processen 'lpsched' som alltid aktiveras så snart någon utskrift finns i kön. 'lpsched' tar utskrifterna ur kön och startar de kommandofiler som hör till respektive logisk skrivare och sänder utskriften till den fysiska skrivare som definierats för den logiska skrivaren. lpsched akriveras även då systemet startas upp för att tömma eventuellt kvarvarande skrivarkö, som finns kvar sedan föregående avstängning.

Ex. 1: Skriv till skrivaren 'main', statuslistan som ges av lpstat:

```
lpstat ö lp -dmain
```

Ex. 2: Skriv filen 'myfile' till skrivaren 'myfilter':

```
lp -dmyfilter myfile
```

Se även spooler-systemet **lpr** enligt nedan. Endast ett av dessa system (**lp** eller **lpr**) skall vara aktiverade i systemet. Båda levereras med operativsystemet och **lp**-systemet är normalt aktiverat. Se **IH**.

Alla konfigurationsdata och status för **lp**-systemet finns i biblioteket /usr/spool/lp.

Övriga kommandon i **lp**-systemet är:

lpstat listar status för begärda utskrifter och även för **lp**-systemet som helhet.

cancel stoppar och tar bort en köad eller pågående utskrift. Som parameter anges ett könummer, enligt listan från **lpstat**, eller ett skrivarnamn.

disable hindrar temporärt alla utskrifter från köerna till en viss skrivare. Kommandot kan även ta bort hela kön av utskrifter till en skrivare.

enable möjliggör att utskrifter kan göras till en viss skrivare.

/usr/lib/lpmove flyttar köade utskrifter från en skrivare till en annan.

/usr/lib/lpshut stoppar alla utskrifter temporärt genom att stoppa processen lpsched.

/usr/lib/reject hindrar **lp** att sända utskrifter till en viss skrivarkö.

/usr/lib/accept tillåter **lp** att skriva till en skrivarkö.

/usr/lib/lpadmin Detta är ett kommando för att skapa eller ändra konfigurationen för **lp**-systemet. Kopplingar införs mellan logiska skrivare, fysiska skrivare och de styrprogram som ska användas.

Hanteringen av **lp**-systemet beskrivs med praktiska råd i kapitel 9 i denna manual och detaljer finns i **AH** för de olika kommandona.

/bin/lpr

Ett alternativt printer-spoolersystem finns även med i grundpaketet. Kommandot **lpr** kan användas att sända utskrifter till "logiska skrivare" i systemet. (jämför **lp**-kommandot). Det är bakgrundsprocessen **/usr/lib/-lpd** som sköter utskriften från den kö som bildas med **lpr**. Notera att endast en av dessa båda system skall vara aktiva is systemet. Se installationsmanualen för den använda datorn.

Med **lpr** finns ett antal optioner, som beskrivs i **AH**. En option (-t) är vilken logisk skrivare som utskriften ska sändas till. Utan angivande av logisk skrivare riktas utskriften till /dev/lp. Det som ska skrivas kan även riktas till **lpr**-kommandot med en pipe, varefter det läggs i skrivarkön.

Ex 1: Skriv filen 'myfile' till skrivaren pr1:

```
lpr -tpr1 myfile
```

Kopplingen mellan "logisk skrivare" och fysisk skrivare görs i filen /usr/lib/lpdpar/lpdtable. Denna fil innehåller en rad för varje skrivare och de villkorssträngar som kan anges i optionen -t till **lpr** för att välja skrivare.

Kommandot **queue** används för att lista vilka utskrifter som ligger i kö för att skrivas ut med **lpr**-spoolersystemet. Med **queue** kan icke påbörjade utskrifter tas bort ur kön. Utskriftsordningen styrs av en utskriftsprioritet som kan ändras med kommandot **queue**.

Hanteringen av **lpr**-systemet beskrivs med praktiska råd i kapitel 9 och detaljer finns i **AH** för de olika kommandona.

.mail.rc

Filen .mail.rc kan finnas i användarens inloggningsbibliotek och innehåller då en lista över alias-namn för kommandot **mail**. Detta är en lista med förkortningar som av **mail** översätts till de verkliga användarnamnen, eventuellt på andra maskiner i en nätverk. **Mail** söker först i denna lokala fil, därefter söker den i motsvarande globala alias-fil som heter /usr/lib/mail/mail.rc.

Varje rad i filen innehåller två fält med alias-namn och den verkliga mail-adressen. Dessutom kan kommentarer fritt läggas in efter dessa två fält.

Exempel: Ett meddelande ska sändas med kommandot:

```
mail karl
```

till Kalle Karlsson på en maskin som heter com2 genom nätverket via en annan maskin som heter com1. Kalle Karlsson har användarnamnet kalle.

Raden i .mail.rc ska då vara:

```
karl        com1!com2!kalle      Kalle Karlsson
```

/usr/lib/mail/mail.rc

Denna fil innehåller globala alias-namn för kommandot **mail**. Kommandot **mail** söker först i användarens egen fil ¤HOME/.mail.rc och sedan i den globala /usr/lib/mail/mail.rc efter det namn som angivits som parameter till **mail**.

Formatet hos filen är samma som för filen .mail.rc i föregående stycke.

/etc/mknod

Kommando för att lägga till nya enhetsnamn i biblioteket /dev. Detta förutsätter att motsvarande drivrutin redan finns i operativsystemet. **/etc/mknod** gör ett nytt enhetsnamn och anger vilken drivrutin och vilken fysisk kanal som skall användas för enheten. Vilka enhetsnummer som normalt används ses i kapitel 15.

Exempel:

Definiera en ny terminalenhet med namnet 'tty06' och kanal nummer 6. Systemets drivrutin för terminaler med seriell anslutning har nummer 1. I kommandot **mknod** anges även att enheten arbetar teckenvis genom tecknet 'c' för 'character oriented'. För inloggning krävs även ändring i /etc/inittab, dvs att terminalen aktiveras.

```
/etc/mknod /dev/tty06 c 1 6
```

/etc/mkuser

Kommando med vilket en privilegierad användare (super-user) kan lägga till en ny användare i systemet. Programmet är interaktivt.

/etc/motd

Detta är en textfil, vars innehåll visas för varje användare efter inloggning på systemet. Den systemansvarige kan här lägga in lämpliga aktuella meddelanden. Textfilen visas endast om ett meddelande finns i filen.

/dev/nvram

Starttabeller i en fast men programmerbar minneskrets (ett sk NVRAM, non-volative RAM). Anger bl a varifrån systemet skall laddas vid start. Värdena i NVRAM kan ändras med **/sas/bootpar** eller **/etc/bootpar**. Se /sas/bootpar i kapitel 3 för detaljer.

/bin/passwd

Kommando med vilket varje användare kan ändra sitt lösenord för inloggning.

En privilegierad användare (super-user) kan även ändra andra användares lösenord med /etc/passwd.

Vid inmatning av lösenord syns inga ekon på bildskärmen. Lösenordet måste ges två gånger för att minska risken för misstag. Ett lösenord ska omfatta minst sex tecken och får inte vara lika med användarnamnet. Se **AH** för mer detaljer.

Om de extra tecknen för tidsbegränsning av lösenordet finns i fält 2 i filen /etc/passwd, kan användaren hindras ändra lösenordet enligt de villkor som beskrivs ovan för fält 2 i /etc/passwd.

Kommandot **passwd** kan, med optionen -f, användas även för att ändra användarens lösenord i filen /etc/d_passwd om denna används i systemet för sekundär inloggning.

/etc/passwd

Lista över användare. Nedan ges ett exempel, varefter de olika fälten förklaras. Nya användare läggs till med kommandot **/etc/mkuser**.

En användare kan även läggas till manuellt genom att editerar in en ny rad i /etc/passwd och i /etc/group. För användare som skall kunna skapa nya filer måste ett bibliotek skapas, som ägs av användaren. Då skall även filen .profile kopieras till det nya biblioteket. Allt detta görs automatiskt av kommandot **/etc/mkuser**.

Exempel på /etc/passwd fil:

På grund av begränsad radlängd i manualen delas långa rader i listningen nedan. På bildskärmen kan de se annorlunda ut.

```
root::0:3:0000-Admin(0000):/:
daemon:*noway*:1:12:0000-Admin(0000):/:
bin:*noway*:2:2:0000-Admin(0000):/bin:
sys:*noway*:3:3:0000-Admin(0000):/usr/src:
adm:*noway*:4:4:0000-Admin(0000):/usr/adm:
uucp:*noway*:5:1:0000-uucp(0000):/usr/lib/uucp:
nuucp::6:1:0000-uucp(0000):/usr/spool/uucppublic:
          /usr/lib/uucp/uucico
sync::20:1:0000-Admin(0000):/:/bin/sync
rje:*noway:68:8:0000-rje(0000):/usr/rje:
shqer:*noway:69:8:0000-rje(0000):/usr/rje:
lp:*noway:71:2:0000-lp(0000):/usr/spool/lp:
who::72:2:0000-who(0000):/:/bin/who
jag:3Uuv4MJme0Khk:110:110:Sven Svensson:/usr/sven:
kalle:rrFbrSmdXJo0s:111:110:Karl Karlsson:/usr/karl:
```

Här har ännu inget lösenord definiterats för 'root', vilket bör ske så snart systemet installerats.

Inloggning med användarnamnen 'sync' och 'who' medför i detta exempel en direkt exekvering av kommandona sync respektive who. Något lösenord behövs ej för dessa då utloggning sker automatiskt efter att kommandot avslutats.

Filen innehåller en rad per användare. Raden har sju fält, separerade med kolon ':'. Endast femte fältet med kommentaren får innehålla mellanslag.

Fält 1:
Användarnamnet. Detta är det namn som anges vid inlogging. Små bokstäver måste användas.

Fält 2:
Krypterad version av lösenordet (password). Inte ens super-user kan lista ut verkliga lösenordet. Ett tomt fält anger att inget lösenord krävs vid inloggning.

Det krypterade lösenordet består av 13 tecken. I position 14 kan finnas ett kommatecken ',' följt av två tecken och en sträng. Kommat och de två tecknen anger att lösenordet skall ha en begränsad giltighet i tiden och editeras in av superuser om användaren ska tvingas ändra sitt lösenord regelbundet.

Första tecket anger maximala tiden i veckor som samma lösenord får användas. En användare som loggar in efter att maximala tiden har gått tvingas ange ett nytt lösenord. Nästa tecken anger minsta antal veckor som ett lösenord ska användas innan det får ändras. De två tecknen anger antal veckor i området 0..63, som (asciivärdet-46), dvs '/' = 1 vecka och 'z' = 63 veckor. Om de två tecknen är '..' tvingas användaren ge nytt lösenord vid nästa inloggning. Om de två tecknen är './' kan endast superuser ändra lösenordet.

Sista strängen lagras automatisk och anger när det aktuella lösenordet definierades.

Fält 3:
Användarnummer. Unikt nummer (ett heltal större än eller lika med noll) för varje användare. 'root' MÅSTE ha användarnummer 0, vilket anger att 'root' är privilegierad (super-user). Vanliga användare bör ha nummer större än 99.

Fält 4:
Användargruppnummer (ett heltal större än eller lika med noll). Vid normal inloggning blir en användare automatiskt inloggad i den grupp som anges här. Grupp 1 används normalt för standardguppen 'other' och vanliga användargrupper bör ha nummer större än 99. Efter inloggning kan användaren byta grupp med kommandot **newgrp**, förutsatt att han känner till gruppens lösenord. Man behöver inte ange gruppens lösenord vid vanlig inloggning. Jämför filen /etc/group.

Fält 5:
Kommentarfält, där förklarande text kan anges. T ex användarens namn eller andra data. Vissa teststrängar som anges här kan användas av kommandot **login** som speciella parametrar. Se **login** i **SA**.

Fält 6:
Inloggningsbibliotek, där användaren automatiskt hamnar vid inloggning och dit **cd**-kommandot utan parametrar leder om detta inte ändrats efter inloggning genom omdefiniering av shellvariabeln HOME.

Fält 7:
Här anges vilken process som skall startas när användaren loggar in. Här kan anges en annan kommandoavkodare eller en vanlig process. Tomt fält anger att standardkommandoavkodaren shell skall användas (**/bin/sh**).

Om en vanlig process anges, kommer användaren endast att kunna utföra den processen och aldrig komma åt systemkommandona om inte den startade processen kan skapa nya processer. Då den angivna processen avslutas loggas användaren automatiskt ut.

Standard IN, UT och FEL-kanaler finns definierade även utan kommandoavkodare.

Ett annat sätt att automatiskt starta ett program vid inloggning är att lägga ett kommando i filen .profile, varvid användaren efter avslutning av detta kommando har tillgång till den vanliga kommandoavkodaren. Om användaren automatiskt skall loggas ut kan kommandot startas med **exec** i .profile.

Om detta fält är enbart '*' och kommandot 'chroot' finns i systemet (ej i grundsystemet) så försöker loginprocessen initiera en sekundär inloggning efter att ha bytt rootfilsystem till inloggningsbiblioteket.

/etc/powerfail

Detta är en kommandofil som automatiskt exekveras då nyckelbrytaren på frontpanelen vrids till läget OFF (ej DS90-11). Alla processer avslutas då och filsystemet stängs på ett kontrollerat sätt. Systemmeddelanden ges till enheten /dev/syscon.

Den exekveras även om en power-fail-insignal detekteras från en ansluten spänningsövervakningskrets.

/etc/powerfail stänger systemet på ungefär samma sätt som kommandot /etc/shutdown.

Efter en fullständig avstängning av operativsystemet stänger bootprogrammet av spänningen, men enbart om frontpanelens nyckel är i läget OFF (ej DS90-11).

/etc/profile

Detta är en gemensam kommandofil som automatiskt bearbetas av kommandoavkodaren efter inloggning till systemet. Den bearbetas före användarens egen .profile. Normalt ska /etc/profile sätta parametern TZ för lokal tid ur filen /etc/timezone och visa eventuellt systemmeddelande ut filen /etc/motd.

I exemplet nedan läggs även systemets nodnamn in i prompten. Nodnamnet kan ändras med kommandot **/etc/mkcfig**. Dessutom visas en lista över aktuella nyhetsmeddelanden ur biblioteket /usr/news om motsvarande kommando finns i systemet. Detta kommando ingår inte i grundsystemet.

Exempel på /etc/profile:

På grund av begränsad radlängd i manualen delas långa rader i listningen nedan. På bildskärmen kan de se annorlunda ut.

```
TZ=`cat /etc/timezone`
if Ä -f /etc/systemid Å
then
  PROMT=`cat /etc/systemid`
fi
if Ä ¤LOGNAME != "adm" -a ¤LOGNAME != "nuucp" -a Ö
     ¤LOGNAME != "sys" Å
then
  if Ä ¤LOGNAME != "root" Å
  then
    if Ä -f /etc/motd Å
    then
      cat /etc/motd
    fi
    PS1=¤äPROMTå'¤ '
    if Ä -x /usr/bin/news Å
    then
      news -n
    fi
  else
    PS1=¤äPROMTå'# '
  fi
else
  PS1=¤äPROMTå'¤ '
fi
export TZ PATH PS1
```

.profile

En kommandofil som ska ligga i varje användares inloggingsbibliotek. Den utförs automatiskt av kommandoavkodaren shell när användaren loggar in på systemet och används ofta för att definiera lämpliga styrtecken för den använda terminalen med **stty** och för att definiera kommandovariabler. Kommandovariabler som ska kunna användas av andra program måste "exporteras" med kommandot '**export**'.

Se även filen /etc/profile. Kommandona i den gemensamma filen /etc/profile exekveras före .profile vid inloggning.

Notera att de terminalstyrtecken som initieras med .profile endast gäller så länge användaren är inloggad. Om man byter användarnamn med kommandot **su**, exekveras emellertid inte den nya användarens .profile om inte bindestreck (-) ges som option till kommandot (**su** -).

I .profile kan kommandon anges så att ett visst program startas automatiskt när användaren loggar in. Om detta program startas med **exec** sker automatiskt utloggning då det startade programmet avslutas. Se även /etc/passwd.

I .profile kan en mängd olika användarspecifika shell-variabler definieras.

BI a kan terminaltypen sättas, t.ex. vt100 eller adm3 för de program som använder shellvariabeln TERM och motsvarande hjälpfiler i biblioteket /usr/lib/terminfo eller i filen /etc/termcap. I dessa filer finns information om styrtecken för olika terminaler, som till exempel markörstyrtecken, styrning av videoattribut m m. Terminfo innehåller definitionerna i kompilerad form, medan /etc/termcap är i textformat. Vissa program, exempelvis editorn **siv**, använder alltid formen /etc/termcap.

I .profile kan TERM t ex definieras enligt nedan för vt00-kompatibla terminaler.

```
TERM=vt100
export TERM
```

Användaren kan även lägga in egna kommandon i .profile, förutsatt att att användaren har läs- och skrivtillstånd i filen. Normalt bör varje användare ha en egen kopia av .profile och inte bara en länk till systemets .profile.

Även standard shellvariabler, som definierats av loginprocessen kan ändras här, till exempel:

HOME=/usr/adam/arbetsfiler betyder att hem-biblioteket ändras från inloggningsbiblioteket som definierades i /etc/passwd.

PATH=:/usr/adam/bin:¤PATH betyder att kommandon som anges utan biblioteksnamn söks först i aktuellt bibliotek, sedan i biblioteket /usr/adam/bin och först därefter i de bibliotek som är systemstandard enligt den definition på PATH som sattes upp vid inloggningen. Första tomma fältet före : anger att det aktuella biblioteket skall sökas först. Som standard söks i /bin och i /usr/bin om PATH inte omdefinieras.

Exempel på en .profile-fil:

```
stty -tabs cr0 ff0 nl0 echoe erase 'ÜH' kill 'ÜX'
TERM=twi72
echo 'Terminaltyp (TERM) är satt till ' ¤TERM
PATH=¤PATH:/usr/ucb
export TERM PATH
```

/etc/rc

Kommandofil, som automatiskt bearbetas när systemet övergår till fleranvändarnivå (systemnivå 2, dvs. bootnivå 3). Den bearbetas även om man med kommandot **telinit 2** ändrar systemnivån till 2. Det är ett kommando i /etc/inittab som startar /etc/rc på systemnivå 2.

I /etc/rc finns t ex kommandon för att rensa bort gamla temporärfiler och för att starta upp speciella processer för initiering av systemet, t ex processer för hantering av printerspoolern.

Generellt bör annars processer som skall finnas permanent i systemet startas med ett eget kommando i /etc/inittab. Detta gäller till exempel nätverksprogram eller speciella filhanterare. Vissa program måste däremot startas i /etc/rc, nämligen program som skapar egna processer och själva avslutas medan de nya processerna lever kvar i systemet. Denna typ av program bör även stoppas med hjälp av särskilda kommandon i shell-kommandofilen /etc/shutdown.

Standard IN, UT eller FEL-kanaler är riktade till enheten /dev/syscon under exekvering av kommandona i /etc/rc. Denna länkas normalt till den fysiska huvudkonsolen /dev/console genom ett kommando i /etc/inittab som exekveras före /etc/rc. Se kapitlet om **shell** i användarhandbok om standard IN/UT/FEL-kanaler.

Nedan finns ett exempel på hur filen /etc/rc kan se ut, men exakta utseendet beror på vilka extra optioner systemet är utrustat med. Rader med tecknet # i början är kommentarrader och innehåller ibland exempel på hur vissa optioner startas. Om dessa ska aktiveras tas då bara kommentarteckent (#) bort.

Exempel på filen /etc/rc:

På grund av begränsad radlängd i manualen delas långa rader i listningen nedan. På bildskärmen kan de se annorlunda ut.

```
TZ=`cat /etc/timezone`
export TZ
set `who -r`
if Ä ¤7 = 2 Å
then
  echo "Dataindustrier DIAB AB "`uname -svr`" Virtual"
  date
  /etc/devnm / ö /etc/setmnt
  rm -rf /tmp/* /usr/tmp/*
```

```
if Ä -r /usr/lib/tc/tcpar4301.tc0 -a -c /dev/tc0 Å
then
 /usr/lib/tc/tcboot -s -d /dev/tc0 /usr/lib/tc/tc4301 Ö
                          /usr/lib/tc/tcpar4301.tc0
fi
if Ä -r /usr/lib/tc/tcpar4301.tc1 -a -c /dev/tc1 Å
then
 /usr/lib/tc/tcboot -s -d /dev/tc1 /usr/lib/tc/tc4301 Ö
                          /usr/lib/tc/tcpar4301.tc1
fi
if Ä -f /usr/lib/cron/log Å
then
  mv /usr/lib/cron/log /usr/lib/cron/OLDlog
  > /usr/lib/cron/log
fi
rm -f /usr/spool/uucp/LCK*
rm -f /usr/adm/acct/nite/lock*
mv /usr/adm/sulog /usr/adm/OLDsulog >/dev/null 2>&1
if Ä "`uname -n`" != '(empty)' Å
then
  uname -n >/etc/systemid
  chmod 644 /etc/systemid
fi
if Ä -x /etc/cron Å
then
  n=`ps lax ö fgrep cron ö wc -l`
  if Ä ¤n = 0 Å
  then
    nice -16 /etc/cron
    echo "CRON started"
  fi
fi
n=`ps lax ö fgrep lpsched ö wc -l`
if Ä ¤n = 0 Å
then
  rm -f /usr/spool/lp/SCHEDLOCK
  nice -20 /usr/lib/lpsched
  echo "LP scheduler started"
fi
# /bin/su - adm -c /usr/lib/acct/startup
```

/etc/rmuser

Kommando med vilket en privilegierad användare (super-user) kan ta bort användare i systemet. Inloggningsbiblioteket kan tas bort automatiskt, liksom gruppdefinitionen. Se **AH** för detaljer.

/etc/setspeed

/etc/setspeed sätter hastigheten eller andra parametrar för olika enheter. Vanligen används setspeed till skrivare. Man kan blanda hastigheter och enheter i uttrycket, som t ex:

```
/etc/setspeed -s2400 /dev/lp -s1200 /dev/letter
```

Detta sätter hastigheten för /dev/lp till 2400 baud och hastigheten för /dev/letter till 1200 baud.

/etc/setspeed kan även ändra övriga kommunikationsparametrar, som då ges på samma format som till kommandot **stty**. Detta sker om optionen -h anges. Notera att i detta fall skall < anges framför enhetsnamnet. Exempelvis om skrivaren kräver 8-bits data och ingen paritet förutom 2400 baud:

```
/etc/setspeed -h 2400 -parity -istrip </dev/lp
```

/etc/setspeed ligger kvar som en "demon". När den tas bort , t ex vid utloggning om enheten är en terminal, upphör de uppsatta hastigheterna att gälla.

Vid fel i setspeed p g a felaktiga parametrar startar ej programmet.

Om **/etc/setspeed** lägges in i /etc/rc för att automatiskt startas då systemet startas, kan det ligga aktivt hela tiden. I /etc/rc skall då kommando-raden med /etc/setspeed avslutas med tecknet **&**, för att tillåta /etc/rc att fortsätta bearbetas även vid fel från /etc/setspeed om t ex en angiven skrivare ej är ansluten.

OPTION

```
-in Sätter upp inhastigheten till n,

-on Sätter upp uthastigheten till n,

-sn Sätter upp in- och uthastigheterna till n,
    där n väljs ur följande hastigheter: 50, 75, 110,
    134, 150, 200, 300, 600, 1200, 1800, 2400, 4800,
    9600, 19200, 38400, exta, extb.

-h  parametrar < enhet
    Med denna option kan alla kommunikationsparametrar
    anges, exakt som för kommandot stty, vilket nämligen
    anropas av setspeed. Mellanslag krävs efter -h.
```

/etc/shutdown

Detta kommando kan användas då operatören vill gå ner i enanvändarmod eller vid manuell initiering av en fullständing avstängning av systemet. Endast användare med namnet 'root' som är inloggade på systemets fysiska huvudkonsol (/dev/console) tillåts ge kommandot **/etc/shutdown**.

Meddelanden sänds till alla inloggande som får en viss maximal tid på sig att avsluta. Därefter avbryts alla processer i systemet. Speciella processer för t ex terminalkoncentratorer eller nätverk stängs av på ett riktigt sätt innan filsystemet stängs av och systemet tas ned.

Om kommandot ges med optionen '-k' stängs systemet helt, tills texten 'System halted' visas på huvudkonsolen.

Kommandot /etc/shutdown beskrivs i detalj i kapitel 2.3.

/etc/shutdown måste alltid användas vid avstängningen av datormodellen DS90-11, medan öriga datorer normalt stängs av genom att vrida nyckeln till OFF.

/bin/stty

/bin/stty är ett kommando som används för att visa eller ändra kommunikationsparametrar för aktiva terminaler. Se **AH** där kommandot och dess parametrar beskrivs i detalj. För återställande av förstörda terminalparametrar används kommandot enligt nedan från den berörda terminalen. Detta kan vara användbart om något program avslutats felaktigt, vilket kan visas sig genom att tecken skrivna på tangentbordet inte ekas på bildskärmen.

CTRL-J ges både före och efter kommandot.

```
<CTRL-J> /bin/stty sane <CTRL-J>
```

Detta kommando återställer en terminal till en rimlig terminaluppsättning. Baudrate påverkas ej av parametern 'sane'.

/dev/syscon

Detta enhetsnamn anger den virtuella systemkonsolen och skall i normala fall vara länkad till samma serieport som systemets fysiska huvudkonsol. Dvs /dev/syscon, /dev/console och /dev/systty är alltså normalt samma serieport.

Den virtuella systemkonsolen används som terminal i enanvändarmod och kan temporärt länkas till en annan terminal. Vid övergång från fleranvändarmod till enanvändarmod, länkas alltid /dev/syscon till den terminal varifrån kommandot **'telinit s'** gavs. I systemet blir detta emellertid alltid huvudkonsolen (/dev/console), eftersom övergången alltid skall ske med kommandot **/etc/shutdown** och alltid från huvudkonsolen, för att säkerställa att olika optionsprogram blir korrekt avslutade. Vid återgång till normal fleranvändarmod (systemnivå 2 i filen /etc/inittab) länkas den virtuella systemkonsolen till den fysiska huvudkonsolen genom ett kommando i /etc/inittab.

Terminalparametrarna för /dev/syscon blir i enanvändarmod satta till de parametrar som finns i filen /etc/ioctl.syscon.

/dev/systty

Detta enhetsnamn är synonymt med /dev/console och skall alltid vara länkade till samma fysiska serieport och anger var systemets huvudkonsol finns.

/bin/telinit

Detta är ett kommando med vilket operatören kan byta systemnivå och tvinga 'init' att gå genom /etc/inittab på nytt med den nya nivån. Därvid kan en annan kombination av processer aktiveras. **telinit** länkar automatiskt /dev/syscon till den terminal från vilken **telinit** givits, men om inte enanvändarmod väljs återställs normalt länkningen av en kommando i /etc/inittab. /bin/telinit är länkat till /etc/init.

Den nya systemnivån anges som parameter till kommandot. Parametern kan vara:

0..6	Byt till ny systemnivå. Standard är 2.
q (eller Q)	Gå genom /etc/inittab igen, men bara om filen ändrats.
a, b, c	Aktivera endast de processer som i /etc/inittab markerats med systemnivå a, b eller c. Detta kommando ändrar INTE systemnivån och påverkar inte övriga processer direkt.
s (eller S)	Byt till enanvändarmod. OBS! Använd normalt istället kommandot '**/etc/shutdown**'.

Med 'q' byts inte systemnivån utan 'init' tvingas gå genom filen /etc/inittab igen, men endast om den ändrats. Med 's' kan byte ske till enanvändarmod, men detta bör i normala fall göras med kommandot **/etc/shutdown** för att vara säker på att alla övriga processer blir korrekt avslutade.

Ett exempel kan ses i beskrivningen av /etc/inittab nedan.

/etc/termcap

Detta är en parameterfil med terminalparametrar. Vid leverans är /etc/-termcap kompatibel med de kompilerade filerna i biblioteket /usr/lib/-terminfo. /etc/termcap används alltid av bl a editorn **siv**. Vilken terminaltyp som används bestäms av shellvariabeln TERM. Denna definieras vanligen i användarens .profile fil.

/etc/termcap är en printbar textfil där parametrarna är uttryckta i klartext.

Vilka terminaltyper som finns definierade i /etc/termcap kan ses genom att lista anbart rubrikraderna med följande kommando. Glöm inte apostroferna kring tecknet ö (lilla ö).

```
fgrep 'ö' /etc/termcap
```

I listan kan raden för adm3a t ex se ut som följer. Rubrikraderna består av fält, begränsade av tecknet ö. Andra fältet (adm3a nedan) används normalt som terminaltyp i shell-variabeln TERM.

```
laö|adm3a|ö3aölsi adm3:Ö
```

Efter rubrikraden för varje terminaltyp följer ett antal rader med koder som gäller just den terminaltypen, exempelvis hur markören ska styras, hur många rader som finns på bildskärmen, koden för att rensa bildskärmen, koder för att styra videoattribut på skärmen m m.

och hela filen kan ses med:

```
cat /etc/termcap
```

/usr/lib/terminfo

Detta bibliotek innehåller parametrar för styrning av de terminaler som kan användas till systemet, t ex markörstyrning, bildskärmens storlek med mera. Informationen finns i kompilerad form och data för varje terminaltyp ligger i en egen fil. Biblioteket /usr/lib/terminfo är indelat i underbibliotek så att alla datafiler för alla terminaltyper som börjar med samma bokstav ligger i samma underbibliotek.

Exempel: Parametrar för terminaltypen twist finns i filen /etc/lib/terminfo/t/twist.

Vilka terminaler som är inkluderade i /usr/lib/terminfo kan listas med ls-kommandot, då filernas namn anger terminaltypen.

```
ls /usr/lib/terminfo/*/*
```

Vid leverans är filen /etc/termcap kompatibel med filerna i /usr/lib/terminfo.

/etc/timezone

Denna fil innehåller en textsträng som läses in av /etc/profile och definierar shellparametern TZ.

TZ innehåller en sträng på tre tecken som indikerar lokal tid i utskriften från kommandot date, samt tidsskilladen i timmar mellan GMT (Greenwich Mean Time) och lokal tid.

I exemplen nedan finns två varianter av teckensträngen, MET = Middle European Time (vintertid) och MDT = Middle european Daylight saving Time (sommartid).

Systemet arbetar alltid i GMT-tid, medan kommandon som **date**, **ls** -l med flera anger lokal tid, utgående från parametern TZ. Om TZ inte finns definierad, används istället den 'Time zone' som definierats i systemets NVRAM med /sas/bootpar.

Vid omställning mellan vintertid och sommartid ska strängen i /etc/timezone ändras genom att ta bort eller lägga till strängen för sommartid, och ändra tidsdifferensen enligt nedan för vinter och sommartid.

Ex.1 Svensk normaltid (-1 timme från GMT):

```
/etc/timezone:    MET-1
```

date visar strängen 'MET'.

Ex.2 Svensk sommartid (-2 timmar från GMT):

```
/etc/timezone:    MET-2MDT
```

date visar strängen 'MDT'.

5

5. Systemvård

5.1 Förebyggande underhåll

Med förebyggande underhåll minskar risken för allvarligare fel i ett datorsystem. Vid varje uppstart av datorn görs automatisk vissa systemtester av hårdvaran och dessutom testas automatiskt de anslutna filsystemen med programmet **fsck**.

Systemoperatören bör regelbundet göra backup av de ingående filsystemen och se till att datorns hårdvarumiljö är bra, att damm inte samlas kring och i datorn och att värme och fuktighet och elektriska störningar inte äventyrar funktionen hos systemet.

Kommandot **fsck** ska endast användas av systemadministratören och endast på skivminnen som inte är "mountade" i systemet. Kommandot skall användas under laddningsprogrammet, i bootnivå 1 efter manuell uppstart, om wichesterskivminnet skall testas.

Skivminne, både disketter och winchesterminne, kan testas med kommandot **fsck**. Normalt förstörs ingen data.

Om systemet upptäcker fel i filstrukturen, rapporteras detta. Med kommandot **fsck** är det möjligt att försöka reparera den filstruktur som är förstörd.

Om kommandot **fsck** hittar filer eller bibliotek som inte är knutna till filsystemet, länkas dessa vid reparationen till biblioteket /lost+found. I detta bibliotek finns det alltså möjligheter att återfinna data som gått förlorad vid reparationen.

Nedan är ett exempel på hur **fsck** kan användas för att testa root-filsystemet på den interna winchesterenheten. Systemet stoppas först och startas upp i manuell mod till laddningsprogrammet. Uppstarten kan här ske från det interna skivminnet.

Se i övrigt kommandot **fsck** i **AH**.

Test av root-filsystemet från laddningsprogrammet

```
>/etc/fsck -rr si(X,Y)      /dev/si32

 Checking /dev/si0
 File system:  root          Volume: si0
 Volume size =   82Mb,
 Block size = 2048,13312 I-nodes (13312 cont.)

 Phase 1 - Check I-nodes
 Phase 2 - Check Pathnames
 Phase 3 - Check Unreferenced files
 Phase 4 - Check Link Counts
 Phase 5 - Check Bitmap

 Total number of files:    662
 Total number of blocks: 41472,  37136 - free blocks
```

/etc/fsck	Kommandot självt.
-rr	Ska alltid anges i stand-alone-mod.
si(X,Y)	Den enhet som filsystemet befinner sig på. Denna ska anges i samma form som vid manuell uppstart från enheten.
Övrig text	Skrivs ut av systemet som en upplysning om vad som kontrolleras.

Följande värden gäller för **si(X,Y)** för de olika modellerna:

DS90 -	10/10E	11	20/21
si(X,Y)	si(16,0)	si(0,0)	si(16,0)

Ett inte mountat filsystem, t ex på en diskett, kan testas från enanvändarmod eller fleranvändarmod, men då ska istället enhetsnamnet anges som argument. För test av ett D-NIX filsystem på diskett ges till exempel följande kommando:

```
/etc/fsck /dev/mf0
```

I kapitel 15 listas de olika enhetsnamn som används i systemet.

5.2 Test av filsystem - fsck

Testprogrammet **fsck** för filsystem är interaktivt och både testar och reparerar filsystemet. **fsck** använder redundant systeminformation i D-NIX systemfiler för att utföra flera olika tester av att filsystemet är korrekt. Om något inte stämmer skrivs det ut och operatören kan välja att låta det vara eller beordra **fsck** att försöka rätta felet.

Permanenta fel i filstrukturen kan uppstå om systemet avbryts medan uppdatering av filstrukturen sker eller om fysiska fel uppstår på det externa minnet. **fsck** kan ofta reparera filsystemet genom att utnyttja information om i vilken ordning operativsystemet skriver ner de olika delarna av strukturen till det fysiska filsystemet.

Nedan beskrivs hur ett D-NIX filsystem normalt uppdateras av operativsystemet, tänkbara orsaker till fel och hur **fsck** eventuellt kan reparera felen. De frågor som programmet kan ställa till operatören behandlas och vad dessa innebär.

Kapitel 5.2.5 listar de olika felmeddelandena och möjliga svar operatören kan ge. Där beskrivs vad varje felmeddelande betyder och eventuella följdfel som kan uppträda.

5.2.1 Allmänt

Innan D-NIX operativsystem laddas in och startas görs normalt en test av att filsystemet är korrekt med **fsck**. Detta hindrar systemet från att skriva ner data till ett filsystem vars struktur redan är felaktig, med oberäkneliga resultat. Om fel upptäcks måste dessa rättas innan filsystemet används.

I det följande beskrivs hur D-NIX filsystem är uppbyggt, hur det kontinuerligt uppdateras och hur filsystemet kan bli felaktigt. Därefter beskrivs hur **fsck** kan försöka att korrigera filsystemet.

5.2.2 Updatering av filsystemet

Varje dag kan hundratals filer skapas, ändras och tas bort. Varje gång en fil ändras uppdaterar D-NIX operativsystem flera delar i den struktur som utgör ett filsystem. Dessa uppdateringar måste alla skrivas ner till skivminnet för att filsystemet ska vara korrekt. För att förstå vad som kan ha hänt då ett permanent fel uppstått i filsystemet är det viktigt att gå genom i vilken ordning de olika uppdateringarna sker. Med användning av denna kunskap kan kommandot **fsck** försöka göra rättelser i filstrukturen.

D-NIX operativsystem innehåller minnesbuffrar med block som ska skrivas till filsystemet. Då något skrivs till filsystemet skrivs först till dessa minnesbuffrar varefter dessa ställt i kö för att skrivas till det fysiska filsystemet, t ex ett skivminne. Fysisk skrivning sker inte förrän motsvarande bufferutrymme behövs till annat ändamål, om systemrutinen flush anropas eller när kommandot **sync** ges. Alla block i minnesbuffrarna skrivs dessutom ner regelbundet, kanske var 30:e eller 60:e sekund.

Det finns fyra delar som uppdateras i filstrukturen. Dessa är inoder, indexblock, datablock (bibliotek och filer) och bitmapblock.

Inoder

En i-nod innehåller den egentliga informationen om en fil. Informationen i en i-nod är typen av fil (bibliotek, data eller specialfil), antalet bibliotekspekare som är länkade till filen, en lista över alla block som tillhör filen, storleken av filen och ägare samt modifieringstider.

I-noden skrivs till filsystemet när filen som motsvarar i-noden "stängs" efter att modifieringar gjorts. Mer exakt ställs de i kö för fysisk skrivning enligt ovan.

Indexblock

Det finns tre typer av indexblock - enkla, dubbla och tredubbla. Ett enkelt indexblock innehåller en lista med pekare till de datablock som ingår i filen. Ett dubbelt indexblock innehåller en lista med pekare till enkla indexblock, som i sin tur listar datablock. Ett tredubbelt indexblock innehåller en lista med pekare till dubbla indexblock, som i sin tur har pekare till enkla indexblock med pekare till datablock.

Indexblock skrivs till filsystemet närhelst de ändrats och markerats klara av operativsystemet. Mer exakt ställs de i en kö enligt ovan för att skrivas ner till det fysiska minnet.

Datablock

Ett datablock innehåller enbart filens datainformation eller bibliotekets poster med ingående filer. Varje post i ett bibliotek består av ett filnamn och numret på filens i-nod.

Datablock skrivs till filsystemet närhelst de ändrats och markerats klara av operativsystemet, men enligt ovan skrivs de inte fysiskt till det externa minnet direkt utan ställs i kö.

Bitmap blocks

Bitmappen innehåller information om alla block på ett fysiskt filsystem som inte används till i-noder, indexblock, datablock eller bitmapblock.

Bitmapblock skrivs till filsystemet närhelst de ändrats och markerats klara av operativsystemet, men enligt ovan skrivs de inte fysiskt till det externa minnet direkt utan ställs i kö.

Det finns även ett särskilt bitmap-block i en fil som kallas badspot. Där finns alla block markerade som har fysiska fel på filsystemet och som alltså aldrig skall användas. Detta block skrivs normalt enbart vid formatering av filsystemet.

5.2.3 Fel i filsystem

Ett filsystem kan bli felaktigt på flera sätt. De vanligaste uppträder om datorsystemet stängs av på felaktigt sätt eller vid hårdvarufel i dator eller skivminnesenheterna.

Felaktig avstängning och uppstart

Filsystemet kan bli felaktigt om en avstängning inte sker på rätt sätt, t ex om sync-rutinen inte anropas innan CPU:n stoppas eller innan ett mountat filsystem görs fysiskt skrivskyddat eller innan ett mountat filssystem slås av eller tas bort (ex. en diskett).

Filsystem kan även bli ytterligare förstörda om något program tillåts skriva till ett filsystem som redan är felaktigt. Detta förstör normalt helt möjligheten att reparera filsystemet.

Hårdvarufel

Om någon del av hårdvaran, dator eller skivminne, är felaktig kan svåra fel uppstå. Vissa kan vara korrigerbara eller lokala, som när ett fysiskt block på en skiva blir oanvändbart. Andra fel kan vara allvarligare som t ex om styrenheten till skivminnet är trasigt.

5.2.4 Att upptäcka och korrigera fel i filsystem

Ett passivt filsystem (inte mountat och utan att något är skrivet till filsystemet som inte ännu fysiskt överförts) kan testas för att upptäcka eventuella avvikelser från den givna strukturen genom att jämföra redundant information i filsystemet. Redundant information läses från filsystemet eller beräknas utifrån andra kända data. Det är viktigt att inga processer tillåts skriva till filsystemet under testen på grund av att **fsck** går genom filerna flera gånger (multi-pass).

Om en felaktighet hittas rapporterar **fsck** detta till operatören som får välja om rättelse ska ske eller ej. Det är även möjligt att välja att alltid rätta utan frågor, genom att ange vissa optioner till **fsck**-kommandot.

I detta delkapitel visas hur felaktigheter kan hittas och eventuella rättelser utföras för i-noder, indexblock, datablock med biblioteksposter och bitmapblock. Dessa rättelser kan utföras av **fsck**.

I-noder

Listan över i-noder är det första som testas, varvid i-nod nummer 1 är den första. I-nod 0 finns inte. Varje i-nod testas vad gäller formatet, typen, antal länkar, block med dubbla referenser, dåliga block och storleken på i-noden.

Format och typ

Varje i-nod innehåller ett typbeteckning (mode). Denna mode beskriver typen och status för i-noden. I-noder finns av fyra olika typer:

* Vanlig fil

* Bibliotek

* Specialfil, blocktyp

* Specialfil, teckentyp

Om en i-nod inte är en av dessa anses den vara felaktig. I-noder kan befinna sig i en av tre stadier - oanvänd, använd eller ingendera. I sista fallet är det fel. En i-nod kan få detta felaktiga status om felaktiga data skrivits till i-nod-listan genom t ex ett hårdvarufel. I detta fall kan **fsck** enbart radera i-noden (filen).

Antal länkar

I varje i-nod finns information om antal länkar, dvs antal pekare med olika filnamn i ett eller flera bibliotek som pekar på samma i-nod. **fsck** kontrollerar att länkantalet stämmer med verkliga antalet pekare som hittas i bibliotekshiearkin, med start från root-biblioteket och nedåt i hela filsystemet.

Om det lagrade länkantalet är skilt från noll och verkliga antalet pekare är noll, betyder det att inget filnamn finns för i-noden. Om det lagrade länkantalet är skilt från noll och inte samma som antalet pekare, betyder det att en bibliotekspost med pekare har lagts till eller tagits bort utan att i-noden uppdaterats.

Om det lagrade länkantalet är skilt från noll och men inga pekare finns, kan **fsck** länka den gömda i-noden till ett temporärt filnamn i systembiblioteket /lost+found. Om länkantalet och antal pekare är olika, men skilda från noll kan **fsck** korrigera det lagrade länkantalet i i-noden.

Block med dubbla referenser

I varje i-nod finns en lista med pekare till datablock eller en lista med pekare till indexblock. Dessa pekar på alla block i filsystemet som tillhör i-noden. **fsck** jämför dessa pekare med en lista över redan upptagna block. Denna lista skapas och utökas eftersom i-noderna gås igenom. Om ett blocknummer redan tillhör en annan i-nod, markeras den som dubbelrefererad. Om dubbelrefererade block hittas söker **fsck** upp den andra i-noden som blocket tillhör och försöker undersöka vilken av i-noderna som är felaktig. Vanligen anses den senast ändrade i-noden som den rätta och den äldsta raderas då. Denna situation kan uppstå bl a om bitmapblocket inte blivit korrekt uppdaterat så att ett block som markerats ledigt även tillhör en fil. När filer med dubbelt refererade block påträffas bör operatören kontrollera om blocken verkligen innehåller data som hör till filen och i annat fall manuellt ta bort motsvarande area ut filen.

En stor mängd dubbelrefererade block i en i-nod kan bero på att ett indexblock inte skrivits ner till det fysiska filsystemet. **fsck** frågar då operatören om båda i-noderna ska raderas.

Dåliga block

I varje i-nod listas direkt eller via indexblock, alla block som tillhör i-noden. **fsck** testar om blocknumret ligger inom de min och maxgränser som gäller för filsystemet. Om blocknumret är utanför gränserna anses det som ett felaktigt blocknummer. Om det finns ett stort antal felaktiga blocknummer i en i-nod kan det bero på att ett indexblock inte skrivits ner till det fysiska filsystemet. **fsck** frågar då operatören om i-noden ska raderas.

Kontroll av filstorleken

Varje i-nod har ett 32-bits (4 bytes) fält som anger antal databytes i den fil (eller bibliotek) som är associerad med i-noden. Denna storlek kontrolleras om den är rimlig. T ex ska biblioteksfilers storlek vara en multipel av 16 och antalet blockpekare ska stämma med filens storlek.

En i-nod för en biblioteksfil i systemet har en viss bit satt i typbeteckningen (mode). Storleken av en sådan fil måste vara en multipel av 16 eftersom varje bibliotekspost är 16 bytes (2 bytes med i-nodnummer och 14 bytes för fil- eller underbiblioteks-namnet).

fsck skriver bara ut en varningstext vid sådana fel i en biblioteksfil eftersom det inte finns tillräcklig information för att kunna korrigera felet.

En grov test av filstorleken erhålls genom att räkna ut hur många block som krävs för att rymma filen och jämföra med antalet block som tillhör i-noden. Antalet bytes divideras med blockstorleken och detta avrundas uppåt. Dessutom adderas ett block för varje indexblock som hör till i-noden. Om det verkliga antalet block skiljer sig från detta beräknade antal, varnar **fsck** för detta.

Indexblock

Indexblock tillhör alltid en i-nod och är egentligen enbart en utökning av listan över datablock som tillhör i-noden.

Fel i indexblock kan vara dubbla referenser till datablock eller felaktiga blocknummer enligt ovan. Därutöver kan indexblocket självt felaktigt anses även tillhöra en annan i-nod.

Datablock

De två möjliga typerna av datablock är vanliga datablock och biblioteksblock. Vanliga datablock innehåller de data som lagrats i filen. Biblioteksblock innehåller biblioteksposter.

fsck kan inte testa vanliga datablock.

Varje biblioteksblock testas vad gäller formatet och storleken. Varje post i biblioteksblocket innehåller filnamn och i-nodnummer till motsvarande fil. I-nodnumret testas och måste peka på en befintlig korrekt i-nod. Inodnumret ska vara inom min/max-gränserna för filsystemet och filerna "." och ".." ska finnas korrekt definierade och peka på i-noder av typen 'bibliotek'. Biblioteksblocket ska själv vara refererat i det överordnade biblioteket i filsystemet.

Om, i en post i ett bibliotek, i-nodnumret pekar på en i-nod som inte finns definierad, tar **fsck** bort denna post. Detta kan bl a uppstå om datablocket som innehöll biblioteksposten skrivits ner till filsystemet medan motsvarande i-nod inte skrivits ner.

Om, i en post i ett bibliotek, i-nodnumret pekar utanför i-nodlistan, kan **fsck** ta bort denna post. Detta kan uppstå om felaktiga data skrivits till bibliotekets datablock.

Första biblioteksposten måste vara i-nodnumret för ".". Dess värde ska vara i-nodnumret till biblioteket egen i-nod.

Andra biblioteksposten måste vara i-nodnumret för "..". Dess värde ska vara i-nodnumret för det bibliotek som utgör förälder-biblioteket (eller till bibliotekets egen i-nod om biblioteket är root-biblioteket).

Om någon av dessa i-nodnummer är felaktiga kan **fsck** ersätta dem med korrekta värden.

fsck testar allmänt filsystemets hiearki och hur filer och bibliotek är sammanlänkade. Om bibliotek eller filer hittas som inte refereras i något bibliotek i filsystemet, kan **fsck** länka in dessa i under systembiblioteket /lost+found. Detta kan bli följden om i-noder skrivs ner till filsystemet trots att motsvarande biblioteks-datablock inte skrivs ner.

Biblioteket /lost+found skapas när filsystemet initieras av kommandot **mkfs**. Detta bibliotek har en fast storlek och kan därför bli fullt vid testning av filsystemet med **fsck**. I detta fall bör filerna i /lost+found flyttas eller tas bort innan **fsck** återstartas.

5.2.5 Fsck felmeddelanden

fsck arbetar i flera faser. Efter initiering, går **fsck** genom filsystemet flera gånger för att rensa filsystemet, testa block och storlekar, filnamn och sökvägar, korsreferenser (länkar), antal länkar och bitmappen (vilken kan byggas upp på nytt).

När ett fel upptäcks reporterar **fsck** detta till operatören, om detta inte kopplats borts genom att välja en option till kommandot. I de fall operatören ska få välja åtgärd skrivs en fråga ut och operatören skall vanligen svara med y (för ja eller yes) eller n (för nej). Detta kapitel förklarar de olika meddelandena, möjliga svar och de fel som orsakar felmeddelandet.

Felen som beskrivs i resten av kapitlet är ordnade i den ordning de hittas av **fsck**, genom de olika faser som **fsck** genomlöper.

fsck kan enbart testa filsystem som inte är mountade (se **mount**-kommandot). För att testa rootfilsystemet måste därför datorn startas upp till stand-alone-systemet utan att D-NIX operativsystem startas. Alternativt kan systemet startas med en diskett som root-filsystem om det fasta skivminnet skall testas.

Flera optioner kan anges till **fsck**-kommandot. Dessa beskrivs i **AM**. Vid exekvering i stand-alone-systemet måste optionen -rr användas, varvid inga frågor ges men **fsck** försöker göra sitt bästa för att reparera filsystemet.

Faserna är:

```
Initiering
Phase 1: Check i-nodes
Phase 2: Check pathnames
Phase 3: Check unreferences files
Phase 4: Check link counts
Phase 4b: Fix system files
Phase 5: Check bit map
Phase 6: Replace bit map
Cleanup message
```

Initiering

Innan en test av filsystemet kan göras, måste vissa tabeller skapas och initieras samt vissa systemfiler öppnas. Denna sektion behandlar meddelanden och fel då detta utförs. Fel kan uppkomma i kommandoraden till **fsck**, på grund av begränsat minnesutrymme, då filer öppnas, på grund av fel i filstatus eller storlek av systemfiler och då temporära filer skas.

Invalid option -X

X är en ej tillåten option till **fsck**. Tillåtna optioner är -V -y -n -s -t och -rr. Vid detta fel avbryts **fsck**. Se beskrivningen av kommandot **fsck** i **AM** för mer detaljer.

File name must follow option -t

Optionen -t måste åtföljas av ett filnamn på en fil som ska användas som temporär arbetsfil. Vid detta fel avbryts **fsck**. Se beskrivningen av kommandot **fsck** i **AM** för mer detaljer.

Cannot get memory

fsck misslyckades i sin begäran till operativsystemet efter virtuellt minne till sina tabeller. Detta kan innebära ett allvarligt problem, som kan kräva insats av servicepersonal. Vid detta fel avbryts **fsck**.

Cannot open /etc/checklist

Standardfilen med en lista över de filsystem som ska testas av **fsck** kan inte öppnas för läsning. **fsck** avbryts vid detta fel. Operatören bör kontrollera att rätt åtkomstmoder angivits för filen så att **fsck** har tillstånd att läsa den.

Can not create FFF

fsck kan inte skapa temporärfilen FFF. Programmet ignorerar då det filsystem som håller på att testas och fortsätter med nästa filsystem. Kontrollera åtkomsttillstånden för filen FFF och/eller biblioteket där filen ska skapas.

Read error at block b (n) - CONTINUE?

Blocket med nummer b (block storlek n) kunde inte läsas.

YES Fortsätt, blocket som inte kunde läsas fylls med nollor.
NO **fsck** avbryts.

Write error at block b (n) - CONTINUE?

Blocket med nummer b (block storlek n) kunde inte skrivas till det fysiska skivminnet.

YES Fortsätt, ignorera felet.
NO **fsck** avbryts.

Search for VDS-block?

Pekaren till VDS-blocket (volym descriptor block) är förstörd. VDS-blocket innehåller viktig strukturinformation om filsystemet utan vilket filsystemet inte kan tolkas. Om detta block inte hittas avbryts **fsck**.

YES **fsck** försöker hitta VDS-blocket. Om det hittas får operatören en fråga (med info om blocknumret) om det accepteras eller ej.
NO **fsck** avbryts.

FATAL ERROR: I-node for i-nodelist is corrupted - try to do a fixup?

I-noden som innehåller pekare till i-nodlistan är förstörd. Om denna

inte kan repareras kommer **fsck** att försöka använda den del av i-nodlistan som ligger direkt i följd i filsystemet. I detta fall kan vissa filer förloras.

YES **fsck** försöker återskapa den förstörda i-noden.
NO Ingen reparation görs, endast den ursprungliga delen av i-nodlistan som ligger direkt i följd i filsystemet används.

Phase 1: Check I-nodes

I denna fas testas i-nodlistan. Fel som kan uppstå är då tabeller för antalet länkar skapas av **fsck**, då inoder utan eller med fel antal länkar hittas, då dubbelreferenser till block hittas, om filstorleken är fel eller i-nodens format är fel.

Invalid index block in inode list - fix?

I-noden som innehåller pekare till block för i-nodlistan innehåller felaktiga indexblock. Detta kan betyda att i-nodlistan delvis är förstörd.

```
YES   I-noden repareras.
NO    Ingen åtgärd.
```

Phase 2: Check Pathnames

I denna fas tas biblioteksreferenser bort som pekar på felaktiga i-noder, vilka ändrats i föregående fas. Här listas fel i i-noden för root-biblioteket eller fel i biblioteksreferenser, t ex i-nodnummer som är för stort eller felaktigt eller om blocknummer är dubblerade eller felaktiga.

Root inode not directory (FIX)

I-noden för root-biblioteket (vanligen i-nod nummer 2) är inte av typen bibliotek.

Möjliga åtgärder anges av operatören:

YES Ändra root-bibliotekets i-nod så att typen blir 'bibliotek'. Om i-nodens datablock verkligen inte är innehåller biblioteksposter, kommer då ett stort antal fel att genereras.
NO fsck avslutas direkt.

Dups/bad in root inode (CONTINUE)

I föregående faser har dubbelrefererade block eller felaktiga blocknummer hittats i i-noden för root-biblioteket (vanligen i-nod nummer 2) i filsystemet.

Möjliga åtgärder anges av operatören:

YES Ignorera felet i i-noden för root-biblioteket och försök fortsätta testa filsystemet. Om i-noden för root-biblioteket är fel blir det nu många följdfel.
NO fsck avslutas direkt.

Unable to read directory - continue?

En i-nod pekar på ett bibliotek som inte kan läsas.

YES fsck fortsätter och ignorerar biblioteket.
NO fsck avslutas direkt.

I - node is out of range - remove?

En biblioteksposten har ett i-nodnummer som är större än i-nodlistan eller noll, vilket är fel.

YES Biblioteksposten tas bort.
NO Felet ignoreras.

Null file name - remove?

En biblioteksposten har ett i-nodnummer men inget filnamn.

```
YES  Biblioteksposten tas bort.
NO   Felet ignoreras.
```

Invalid '.'-link - fix?

Posten "." i en biblioteksposten pekar inte till biblioteket självt.

```
YES  Posten ändras att peka på biblioteket självt.
NO   Felet ignoreras.
```

Invalid '..'-link - fix?

Posten ".." i en biblioteksposten pekar inte till förälder-biblioteket.

```
YES  Posten ändras att peka på förälderbiblioteket.
NO   Felet ignoreras.
```

Multiple LINKS to this DIR - remove?

En biblioteksposten pekar på ett bibliotek som redan har en annan förälder.

```
YES  Posten tar bort.
NO   Felet ignoreras.
```

Invalid directory size - fix?

Storleken av ett bibliotek är inte en multipel av 16 bytes.

```
YES  Storleken justeras.
NO   Felet ignoreras.
```

Unknown file type - remove?

Ordet mode i i-noden, som ska ange filtypen, är felaktigt.

```
YES  I-noden tas bort.
NO   Felet ignoreras.
```

Blocks not aligned - remove?

Indexpekare som pekar på tillhörande block är inte korrekt enligt de blockgränser som gäller i filsystemet.

```
YES  Inoden tas bort.
NO   Felet ignoreras.
```

Blocks out of range - remove?

Indexpekare som pekar på tillhörande block pekar utanför fysiska arean för filsystemet.

```
YES  I-noden tas bort.
NO   Felet ignoreras.
```

Invalid size of file - remove?

Filstorleken stämmer inte mot antalet tillhörande block.

```
YES  I-noden tas bort.
NO   Felet ignoreras.
```

Invalid index pointers - remove?

Indexpekaren som pekar på tillhörande block är felaktig.

```
YES  I-noden tas bort.
NO   Felet ignoreras.
```

Blocks claimed by other inode - remove?

Block som tillhör denna i-nod tillhör även en annan i-nod.

```
YES  I-noden tas bort.
NO   Felet ignoreras.
```

I-node not allocated - remove?

En bibliotekspost pekar på en i-nod som är markerad ledig.

```
YES  Posten tas bort.
NO   Felet ignoreras.
```

Unable to read inode - remove?

En bibliotekspost pekar på en i-nod som inte kan läsas.

```
YES  Posten tas bort.
NO   Felet ignoreras.
```

I-node has been deleted - remove?

En bibliotekspost pekar på en i-nod som har raderats av **fsck**.

```
YES  Posten tas bort.
NO   Felet ignoreras.
```

'.'-link missing - fix?

Biblioteket innehåller ingen länk till ".".

```
YES  Lägg till denna post till biblioteket.
NO   Felet ignoreras.
```

'..'-link missing - fix?

Biblioteket innehåller ingen länk till "..".

```
YES  Lägg till denna post till biblioteket.
NO   Felet ignoreras.
```

Phase 3: Check Unreferenced Files

I denna fas testas biblioteksstrukturen ytterligare och fel listas om orefererade bibliotek hittas eller om om referensen till ett föräldra-bibliotek är felaktigt.

Father directory: I-node= n - reconnect?

Biblioteket har ett förälderbibliotek med i-noden n och typen 'bibliotek', men det finns ingen referens till biblioteket från detta förälderbibliotek.

YES Den saknade referensen läggs in i förälderbiblioteket med ett temporärt filnamn.
NO Förälderbiblioteket ändras inte, men operatören kan välja att lägga en referens i systembiblioteket /lost+found istället.

Connect to dir '/lost+found'?

Filen eller biblioteket har inget förälderbibliotek eller biblioteket lades inte in i förälderbiblioteket enligt ovan. /lost+found är ett bibliotek med fast storlek, varför det kan bli fullt. I detta fall måste filerna i /lost+found gås igenom och flyttas eller tas bort innan **fsck** startas igen.

YES Filen eller biblioteket läggs in i biblioteket /lost+found.
NO Ingen åtgärd.

Remove file?

Filen eller biblioteket har inget förälderbibliotek och lades inte in i /lost+found enligt ovan.

YES Filen raderas och i-noden nollställs.
NO Ingen åtgärd.

Phase 4: Check Link Counts

I denna fas genomgås länkningsinformationen efter att eventuella korrigeringar enligt fas 2 och 3 gjorts. Felen listade nedan är från orefererade i-noder där inte en referens skapats i föregående fas, fel i länkantalet för filer, bibliotek eller specialfiler eller felaktiga block i filer eller bibliotek.

Invalid link count, I-node= n
Is x, should be y - adjust?

I-nodens länkantal stämmer ej överens med antal referenser till i-noden.

```
YES  Länkantalet rättas till.
NO   Ingen åtgärd.
```

Inode = n, is ok but has no father - remove?

I-noden är en korrekt i-nod, men ingen referens till den har hittats och den har inte lagts in i /lost+found i en tidigare fas.

```
YES  I-noden och därmed filen raderas.
NO   Ingen åtgärd.
```

Inode is not referenced - remove?

I-noden är en korrekt i-nod, som saknar referenser. Dessutom är i-nodens länkantal noll.

```
YES  I-noden och därmed filen raderas.
NO   Ingen åtgärd.
```

Phase 4b: Fix system files

Inode for I-nodelist was in error and corrected
Update inode?

I-noden med pekare till resten av i-nodlistan var felaktig men **fsck** bör kunna återställa den.

```
YES  I-noden uppdateras.
NO   Ingen åtgärd.
```

Fix up cont. part of I-nodelist?

I-noden med pekare till resten av i-nodlistan var felaktig. **fsck** kan endast återställa den del av listan som lagrats i direkt på varandra följande block.

```
YES  I-noden uppdateras.
NO   Ingen åtgärd.
```

I-node for bitmap was in error - fixup?

I-noden som pekar ut bitmapblocken är felaktig.

```
YES  I-noden korrigeras.
NO   Ingen åtgärd.
```

I-node for root directory was in error - fixup?

I-noden som pekar ut root-biblioteket var felaktig.

```
YES  I-noden korrigeras.
NO   Ingen åtgärd.
```

I-node for badspot file was in error - fixup?

I-noden som pekar ut filen 'badspot' med en bitmap över fysiskt felaktiga block i filsystemet, var felaktig.

```
YES  I-noden korrigeras.
NO   Ingen åtgärd.
```

Phase 5: Check bit-map

I denna fas testas bitmappen. Fel kan uppstå på grund av felaktiga block i bitmapfilen, av att block som används eller är oanvända markerats felaktigt i bitmappen eller av att totala antalet block i bitmappen inte stämmer med filsystemet fysiska storlek.

Dessa tester utförs inte om **fsck** opionen -rr angivits, varvid den fysiska bitmappen helt ignoreras och återskapas av **fsck**.

Phase 6: Replace bit map

fsck har nu skapat en intern bitmap efter genomgång av och eventuell reparation av hela filsystemet. Om denna beräknade bitmap inte överensstämmer med den fysiska bitmappen kan den rättade bitmappen skrivas ner till filsystemet.

Replace bit map?

Bitmappen som **fsck** beräknat i minnet stämmer ej med bitmappen på det fysiska skivminnet.

```
YES   Bitmappen på skivminnet uppdateras.
NO    Ingen åtgärd.
```

Cleanup

*******FILE SYSTEM WAS MODIFIED*******

Detta är en meddelande som **fsck** ger då korrigering gjorts av filsystemet.

5.3 Konfiguration D-NIX - mkcfig

Det finns möjlighet att ändra vissa parametrar i operativsystemet för optimering av systemet till den aktuella tillämpningen. Dessa parametrar läses in vid uppstart av systemet.

Detta behöver emellertid normalt inte göras annat än i undantagsfall, till exempel om ovanligt många samtidiga processer behövs eller vid ovanligt intensiv användning av stora och många filer. Om behov uppstår, till exempel på grund av felmeddelanden från systemet, bör först programmen kontrolleras. Onödigt stora parametervärden (särskilt för filbufferarean) tar plats i primärminnet, medan för små värden kan göra systemet långsammare i vissa fall. Om systemet har ett stort primärminne, bör detta i första hand utnyttjas genom att använda en stor area för filbuffertar, om snabb exekvering är viktig.

Kommandot **mkcfig** används för att ändra i ett parameterblock i operativsystemfilen på skivminnet. **mkcfig** är interaktivt, men kan även startas med optioner och parametrar för direkt ändring av systemparametrarna. Då mkcfig startas interaktivt, visas alla systemparametrarna på skärmen enligt exemplet nedan, varefter operatören eventuellt ändrar och skriver tillbaka blocket. Därefter kan systemet startas om, efter normal avstängning, för att de nya parametrarna skall gälla.

Fler parametrar kan tillkomma i senare versioner.

Endast super-user kan använda mkcfig och bör befinna sig i root-biblioteket, där normalt operativsystemfilen finns. Gör gärna en backup-kopia av filen innan den ändras.

Kommandot startas i biblioteket "/" med

```
/etc/mkcfig dnix
```

där 'dnix' är det normalt använda namnet för operativsystemfilen. Lista root-biblioteket för att hitta rätt namn. Texten som mkcfig skriver är på svenska eller engelska, beroende på om variabeln LANGUAGE sätts till 'swedish' eller 'english' innan **mkcfig** startas. Exempel:

```
LANGUAGE=swedish
export LANGUAGE
/etc/mkcfig dnix
```

Nedan är ett exempel på en utskrift från mkcfig.:

Konfigureringsparametrar för /dnix

```
    Max antal aktiva processer.....:   64
    Max antal laddade filer........:   64
    Max antal noder................:  256
    Filbufferstorlek i lk-block....:   64
    Max antal öppna filer..........:   96
    Max antal fillås...............:  128
    Max antal 'mountade' filsystem.:    4
    Maxstorlek för 'page swap area' i Mb.....:    8
    Max antal meddelande köer......:   20
    Max antal semaforer............:   20
    Max antal semafor-justering-vid-exit.....:  128
    Max antal delbara minnes segment.....:   20
    Max antal invalda minnes segment.....:  128
    Antal sekunder mellan filsystemuppdatering...:   60
    Antal tecken i tty input buffer.....:  255
    Nodnamn.....................:  ''

    Vill du ändra konfigureringsparametrar?
    ("j" = ja, "a" = avbryt, övrigt = nej) ...:___
```

a för att avbryta utan ändringar!
j för att ändra. Då ger programmet frågor interaktivt. Möjliga värden anges i frågan. Svara bara RETURN om de gamla värden ska behållas.
RETURN eller nej efter att ha gjort ändringar, varvid en ny fråga kommer om parametrarna ska skrivas till operativsystemsfilen, som angivits i mkcfig-kommandot.

```
    Vill du verkligen ändra parametrarna för dnix?
    ("j" = ja, "n" = nej) ...:___
```

j för att uppdatera filen.
n för att avsluta UTAN uppdatering.

EXEMPEL: Ändra 'Max antal noder' till 512.

```
Vill du ändra konfigureringsparametrar?
("j" = ja, "a" = avbryt, övrigt = nej) ...:j

Max antal aktiva processer.....................:   64
nytt värde (mellan 10 och 2048)? ___          (RETURN)
Max antal laddade filer.......................:   64
nytt värde (mellan 10 och 2046)? ___          (RETURN)
Max antal noder...............................:  256
nytt värde (mellan 50 och 8192)? 512 RETURN
```

Svara **RETURN** på övriga frågor. Efter alla frågor visas igen alla parametrar, liksom i början samt samma slutfråga.

```
Vill du ändra konfigureringsparametrar?
("j" = ja, "a" = avbryt, övrigt = nej) ...:RETURN

Vill du verkligen ändra parametrarna för dnix?
("j" = ja, "n" = nej) ...:j
```

Nu skrivs de nya parametrarna ut till filen 'dnix' och mkcfig avslutas.

De olika parametrarna beskrivs nedan:

```
Max antal aktiva processer.....................:   64
```

Detta är antal TCB som reserveras i systemet (vardera ca 250 bytes). Aktiva processer är de som listas med kommandot 'ps'. Normalt räcker standardvärdet, men om felmeddelande "End-of-TCB's" kommer, kan det bero på att ett större värde behövs. Kontrollera emellertid först att inte något program felaktigt har skapat för många processer.

```
Max antal laddade filer.......................:   64
```

Detta är max antal processer som använder olika kod-areor i minnet. I kommandot 'ps' kan ses att flera processer ofta har laddats från samma fil, bl.a shell (-sh). Dessa räknas då som en enda laddad fil. Standardvärdet bör räcka i alla fall för denna parameter. Varje block tar ca. 30 bytes.

```
Max antal noder.............................:   256
```

Detta är totala antal noder i systemet, som ligger i en pool och delas av alla processer. Noder används enligt tabellen nedan och varje nod kräver ca 100 bytes. Om felmeddelandet "End-of-Nodes" kommer kan parametervärdet behöva ökas. Kontrollera först att det inte är något programfel. Fler noder kan behövas bl.a. om antal filbuffertar ökas genom att parametern för filbufferstorleken ökas. (Se nedan).

1 nod för varje öppen fil
1 eller flera noder för varje I/O-beställning till en buffrad extern enhet, t.ex. ett skivminne. Flera noder behövs då read-ahead används.
1 nod för varje NoWait-beställning från en process. Detta är inte så vanligt bland normala kommandon.
1 nod för varje I/O-beställning till en I/O-hanterare, vilken i sin tur behöver noder för sina I/O-beställingar.

```
Filbufferstorlek i lk-block.................:    64
```

Detta är systemets total area i Kbytes för filbuffertar. Systemet allokerar buffertar efter behov upp till denna maximala area. Med stor bufferarea behålls fler skivsektorer i bufferarean och färre fysiska skivaccesser krävs, vilket kan minska exekveringstiden för vissa program. Å andra sidan krävs mer primärminne vilket kan orsaka högre swap-frekvens om många eller stora program körs. Process-swapping sker alltid via fysiska skivaccesser.

```
Max antal öppna filer.......................:    96
```

Detta är max antal samtidigt öppna filer, globalt i systemet. Generellt gäller ytterligare en begränsning till max 20 öppna filer per process i D-NIX. Denna parameter kan möjligen behöva ökas vid tillämpningar med ovanligt många samtidigt öppna filer. Felmeddelandet "File-Table-Overflow" ges om parametern inte räcker till.

```
Max antal fillås............................:   128
```

Systemparameter, som anger hur många samtidiga dataareor i systemet som totalt kan hållas låsta. Med D-NIX anrop kan en eller flera areor i en eller flera filer låsas mot access från andra användare, medan uppdatering sker från en process.

```
Max antal 'mountade' filsystem..............:    4
```

Detta är max totalt antal samtidigt 'mountade' filesystem inklusive root-filsystemet, dvs. antalet externa skivminnen.

```
Maxstorlek för 'page swap area' i Mb.....:  8
```

Detta är den maximala sammanlagda virtuella minnesarean i Mbytes för samtliga aktiva processer, eftersom denna parameter bestämmer max antal poster i minnestabellerna. Lagringsutrymme på skivminnet för 'swapping' tas i anspråk dynamiskt, endast då det behövs när primärminnet är fullt. Den verkligt utnyttjade arean kan grovt uppskattas genom att addera MEM_SIZE för alla processer enligt kommandot '**ps -xl**'. Notera också att endast de delar av en programfil som verkligen används laddas in, varför verkliga minnesbehovet kan vara betydligt mindre än summan av filstorlekarna. Felmeddelandet "End-of-Swap area" ges då virtuella minnet är fullt.

```
Max antal meddelande köer.....:  20
```

Systemparameter, som maximerar antal "message queues" i systemet, vilka fungerar som en slags pipes mellan processer internt i systemet. Dessa används sällan i vanliga program.

```
Max antal semaforer.....:  20
```

Systemparameter, som maximerar antal semaforer i systemet. Dessa används sällan av vanliga program.

```
Max antal semafor-justering-vid-exit.....:  128
```

Systemparameter, som anger max antal systemanrop med begäran att systemet automatiskt skall återställa semaforer om en process avbryts utan att återställa på normalt sätt. Detta används sällan av vanliga program.

```
Max antal delbara minnes segment.....:  20
```

Systemparameter som anger max antal globala namngivna fria minnessegment som kan delas av flera processer, dvs. där gemensamma data för flera processer kan lagras.

```
Max antal invalda minnes segment.....:  128
```

Systemparameter som anger hur många inmappningar som samtidigt kan förekomma i systemet mellan processer och fria minnessegment. En process kan mappa in ett eller flera segment i sitt logiska minne för access.

```
Antal sekunder mellan filbufferuppdateringar.:  60
```

Detta anger tiden i sekunder mellan de automatiska uppdateringarna av skivminnena från filbuffertarna. (Jämför kommandot 'sync').

```
Storlek på terminalinputbuffer i bytes.....:  255
```

Detta är storleken på input-buffern för type-ahead som används av terminalrutinen. Den kan behöva ökas om kommunikationslinjer används där mer än 256 byte sänds per block.

```
Nodnamn.....: 'mg'
```

Detta är nodnamnet som identifierar systemet i ett nätverk. Namnet är en sträng med maximalt 8 tecken. Apostroferna kring strängen, som visas av mkcfig, ingår inte. Nodnamnet kan läsas med kommandot '**uname -n**'.

Mkcfig finns med både engelsk och svensk hjälptext.Nedan följer konfigurationsparametrarna för de olika datorerna i DS90-familjen med de engelska texterna:

Konfigurationsparametrar för /dnix DS90-10

```
Max number of active processes.....: 64
Max number of loaded core files.....: 64
Max number of nodes.....: 256
Size of file buffers in lk-block.....: 64
Max number of open files.....: 96
Max number of file locks.....: 128
Max number of mounted file systems.....: 4
Max size of page swap area in Mb.....: 8
Max number of message queues.....: 20
Max number of semaphores.....: 20
Max number of semaphore adjust operations.....: 128
Max number of shared memory segments.....: 20
Max number of attached memory segments.....: 128
Number of seconds between file system sync..... : 60
Number of characters in tty input buffer.....: 255
Node name.....: ''
```

Konfigurationsparametrar för /dnix DS90-10E

```
Max number of active processes.....: 64
Max number of loaded core files.....: 64
Max number of nodes.....: 256
Size of file buffers in lk-block.....: 64
Max number of open files.....: 96
Max number of file locks.....: 128
Max number of mounted file systems.....: 4
Max size of page swap area in Mb.....: 8
Max number of message queues.....: 20
Max number of semaphores.....: 20
Max number of semaphore adjust operations.....: 128
Max number of shared memory segments.....: 20
Max number of attached memory segments.....: 128
Number of seconds between file system sync..... : 60
Number of characters in tty input buffer.....: 255
Node name.....: ''
```

Konfigurationsparametrar för /dnix DS90-11

```
Max number of active processes.................: 64
Max number of loaded core files................: 64
Max number of nodes............................: 256
Size of file buffers in 1k-block...............: 64
Max number of open files.......................: 96
Max number of file locks.......................: 128
Max number of mounted file systems.............: 4
Max size of page swap area in Mb...............: 8
Max number of message queues...................: 20
Max number of semaphores.......................: 20
Max number of semaphore adjust operations......: 128
Max number of shared memory segments...........: 20
Max number of attached memory segments.........: 128
Number of seconds between file system sync..... : 60
Number of characters in tty input buffer.......: 255
Node name......................................: ''
```

Konfigurationsparametrar för /dnix DS90-20

```
Max number of active processes.................: 96
Max number of loaded core files................: 96
Max number of nodes............................: 512
Size of file buffers in 1k-block...............: 256
Max number of open files.......................: 256
Max number of file locks.......................: 128
Max number of mounted file systems.............: 4
Max size of page swap area in Mb...............: 8
Max number of message queues...................: 20
Max number of semaphores.......................: 20
Max number of semaphore adjust operations......: 128
Max number of shared memory segments...........: 20
Max number of attached memory segments.........: 128
Number of seconds between file system sync..... : 60
Number of characters in tty input buffer.......: 255
Node name......................................: ''
```

Konfigurationsparametrar för /dnix DS90-21 (en CPU)

```
Max number of active processes.................: 96
Max number of loaded core files................: 96
Max number of nodes............................: 512
Size of file buffers in lk-block...............: 256
Max number of open files.......................: 256
Max number of file locks.......................: 128
Max number of mounted file systems.............: 4
Max size of page swap area in Mb...............: 8
Max number of message queues...................: 20
Max number of semaphores.......................: 20
Max number of semaphore adjust operations......: 128
Max number of shared memory segments...........: 20
Max number of attached memory segments.........: 128
Number of seconds between file system sync..... : 60
Number of characters in tty input buffer.......: 255
Node name......................................: ''
```

```
/etc/mkcefig: not found
hult# /etc/mkcfig dnix
```

Endrad ds 90-00

```
Configuration parameters for dnix

Max number of active processes.............:  255
Max number of loaded core files............:  255
Max number of nodes........................:  1024
Size of file buffers in Kb.................:  64
Max number of open files...................:  512
Max number of file locks...................:  512
Max number of mounted file systems.........:  4
Max size of page swap area in Mb...........:  8
Max number of message queues...............:  20
Max number of semaphores...................:  20
Max number of semaphore adjust operations..:  128
Max number of shared memory segments.......:  20
Max number of attached memory segments.....:  128
Number of seconds between file system sync..:  60
Number of characters in tty input buffer...:  255
Node name..................................:  'hult'

Do you want to change configuration parameters?
        ("y" = yes, "q" = quit, other = continue) ...:
```

/
~~mia~~# /etc/mkcfig dnix orginal ifpc1 DS90 00

Configuration parameters for dnix

```
Max number of active processes..............:  64
Max number of loaded core files.............:  64
Max number of nodes.........................:  256
Size of file buffers in Kb..................:  64
Max number of open files....................:  96
Max number of file locks....................:  128
Max number of mounted file systems..........:  4
Max size of page swap area in Mb............:  8
Max number of message queues................:  20
Max number of semaphores....................:  20
Max number of semaphore adjust operations....:  128
Max number of shared memory segments........:  20
Max number of attached memory segments......:  128
Number of seconds between file system sync...:  60
Number of characters in tty input buffer.....:  255
Node name...................................:  'mia'

Do you want to change configuration parameters?
        ("y" = yes, "q" = quit, other = continue) ...:
```

6

6. Användarens miljö

6.1 Användarmiljö - (Environment)

Inlogging:
Då datorsystemet är uppstartat kan användare logga in från alla aktiverade terminaler. På dessa terminaler har då systemet skrivit ut texten 'Login: ' eller motsvarande. Om terminalen har varit avslagen kan användaren trycka på RETURN för att få denna frågetext. Texten kan föregås av en systemidentifikation innan Login: skrivs ut. Därefter anges användarnamnet, varefter systemet frågar efter lösenord. Tecknen ekas inte till bildskärmen medan lösenordet skrivs. Medan användarnamn och lösenord skrivs kan inte de normala tangenterna för att backa och sudda tecken användas (Tecken # kan användas för att ta bort föregående tecken men det är inte säkert att det syns på bildskärmen).

Efter godkänd inloggning
Efter inloggning startas en kommandoavkodare (shell), som först skriver ut ett systemmeddelande, som systemoperatören lagrat i filen /etc/motd. Därefter bearbetas de kommandon som användaren har i sin fil .profile. Först efter detta kan användaren börja ge kommandon från tangentbordet. I vissa fall kan i filen .profile finnas kommandon så att ett applikationsprogram automatiskt startas upp då användaren loggar in.

Användaren befinner sig i sitt inloggningsbibliotek efter inloggning, om inte kommandon i .profile givits för att byta bibliotek.

Användaren tillhör en grupp användare enligt de definitioner som finns i filen /etc/passwd och /etc/group. Detta ger honom/henne möjlighet att läsa och skriva filer som inte är egna, men som tillhör gruppen, beroende på vilken behörighet som gäller för de olika filerna.

Användaren kan byta till en annan grupp med kommandot **newgrp** om han känner till lösenordet för den nya gruppen.

Begränsad shell (restricted shell)
Det finns en möjlighet att i filen /etc/passwd definiera att användaren enbart skall få använda vissa kommandon och hindras att skriva till filer utanför det bibliotek där han befinner sig efter inlogging. Då startas en begränsad kommandoavkodare, som inför följande begränsningar:

* Användaren kan inte byta bibliotek.

* Användaren kan inte ändra variabeln PATH.

* Användaren kan inte komma åt filer utanför det aktuella biblioteket genom att / inte får förekomma i filnamn.

* Användaren får inte använda omdirigering av utmatning från kommandon med > eller >>.

Om däremot en vanlig kommandoavkodare (shell) kan startas, dvs finns i de bibliotek som anges i PATH, så finns inga begränsningar där. Likaså kan de kommandon som startas komma åt filer fritt utan begränsningar, men inte genom standard input/output från den begränsade shell.

Shell-variabler:
I kommandoavkodaren kan olika variabler definieras som kan påverka hur kommandon utförs. Alla dessa variabler kallas användarens miljö (environment). Dessa kan ses med kommandot **set**, givet utan parametrar.

Där ingår bland annat följande variabler som automatisk definieras vid inloggning, men användaren kan själv ändra eller lägga till variabler.

```
LOGNAME=användarnamnen för användaren
HOME=användaren inloggningbibliotek
PATH=:/bin:/usr/bin  om den inte omdefineras i .profile
SHELL=namnet på kommandoavkodaren angiven i /etc/passwd
MAIL=den fil där meddelanden kommer med mail
TZ=tidszonen vid tidsangivelser
PS1=¤      (Den prompt som skrivs på bildskärmen)
```

Shellvariabeln **TERM** defineras varnligen i .profile och skall ange namnet på den terminaltyp som används. Den används inte av de enklaste kommandona i systemet, men är nödvändig för alla kommandon som behöver informaion om hur bildskärmen skall hanteras, t ex ordbehandlare som måste veta vilka styrtecken som skall användas.

Då ett kommando skrivs in på tangentbordet kan vissa tecken användas för att ta bort tecken eller hela raden och vissa tecken används för att avsluta inmatning. Vilka dessa är defineras med kommmandot **stty** i filen .profile och gäller alltså först efter en godkänd inloggning. Dessa är följande och är normalt definerade enligt nedan.

Backspace (Backa och sudda föregående tecken)	**CTRL-H**
Delete (Sudda hela raden)	**CTRL-X**
End-Of-File (Logga ut från systemet eller avsluta inmatning)	**CTRL-D**

Notera här att eventuell bakåtpil på terminalen ofta ger helt annan kod till datorn och inte kan användas vid inmatning av ett kommando.

Skrivarsystem lp eller lpr

Två olika skrivarsystem levereras med systemet och systemoperatören aktiverar enbart ett av dessa. Vid utskrift till skrivare kan alltså antingen kommandot **lp** eller kommandot **lpr** användas.

För att underlätta vid utskrifter finns ett kommando som heter **print**, vilket är en liten shell-procedur, som delar in det som skrivs ut i sidor, lägger till sidrubriker och som ska vara anpassad till det aktiva skrivarsystemet. Kommandot **print** anropar kommandot **pr** för formatering och kan därför inte användas då texten skall formateras på annat sätt.

Se kapitel 9 för en fullständig beskrivning av skrivarsystemen.

6.2 .profile

Filen .profile skall ligga i användarens inloggningsbibliotek och innehåller kommandon som utförs direkt efter inloggning. Bl a finns normalt kommandot **stty** för att definera lämpliga styrkoder för kommunikationen mellan dator och terminal. Shell variablen TERM defineras alltid här och användaren kan definiera egna variabler. Dessa skall alltid markeras med **export** för att kunna användas av andra kommandon.

Exempel på filen .profile:

```
stty -tabs cr0 ff0 nl0 echoe erase 'ÜH' kill 'ÜX'
TERM=twist
echo 'Terminaltypen (TERM) är : ' ¤TERM
PATH=¤PATH:/usr/kalle/bin
export TERM PATH
```

I .profile kan även tillämpningsprogram startas upp automatiskt, t ex programmet menu (se D-MENU) som beskrivs i följande avsnitt.

6.3 D-MENU

Principen med D-MENU är att användaren vid inloggning till systemet möts av en meny. Menyn kan antingen vara en av systemets standardmenyer eller en av användarens egna menyer.

Utifrån denna meny kan användaren starta upp olika program i systemet, endera från menyn som visades vid inloggning eller från någon annan meny som kan startas upp från den första menyn.

Alla som startar upp D-MENU behörighetskontrolleras dels av menysystemets egen behörighetskontroll dels av operativsystemets normala behörighetskontroller. En kontroll av att användaren är tillåten att, via den använda terminalen, få tillgång till systemet görs också.

Eventuella fel som inträffar när D-MENU används detekteras och presenteras i klarspråk för användaren. Det gäller både fel i menyhanteraren och fel som uppstår i program och system anrop som startas från menyhanteraren.

Menysystemet innehåller även en menyeditor. Menyeditorn underlättar för användaren att skapa och uppdatera menyer. Både menyhanteraren och menyeditorn är språkoberoende. Detta gäller alla texter i systemet, felmeddelanden, dialogtexter och hjälpfiler.

6.3.1 Allmänt om D-MENU

Väl installerad är D-MENU en allmän användarresurs som binder ihop användaren med alla andra resurser i systemet på ett för användaren enkelt sätt. Användaren har möjligheter att bygga upp menyer och grupper av menyer inom olika användningsområden. D-MENU består av två program:

`/usr/bin/menu`	`program för menyhantering`
`/usr/bin/menuedit`	`program för menyeditering`

D-MENU innehåller dessutom ett behörighetskontroll-system som antingen tillåter eller omöjliggör för användaren att utföra ett val beroende på användarens definierade behörighetsnycklar.

P g a D-MENU's natur är det viktigt att System Administratören ensam ansvarar för följande administrativa uppgifter:

* installation av mjukvara
* skapa användare till menysystemet
* tilldela menyanvändarna behörighetsnycklar
* initialisering av användarnas environment variabler
* skapa och uppdatera systemmenyer.

Den person som fungerar som menyadministratör måste vara prilivigerade som superuser eller kunna logga in som root.

6.3.2 Krav på system och terminaler

Maskinvarubehovet för D-MENU inkluderar följande:

* DS90 dator
* Terminaler

Terminaler som används för D-MENU måste ha en minimumstorlek på skärmen som är 24 linjer med 80 kolumner, mjukvarukontrollerad skärmrensning och cursorpositionering. Terminaltypen måste också finnas definierad i operativsystemets termcap fil.

6.3.3 Ingående programdelar

D-MENU innehåller följande bibliotek och filer.

Bibliotek:

/usr/lib/menues	Bibliotek för D-MENU's systemmenyer och textfiler.
/usr/lib/menues/help	Systemmenyernas hjälpfiler.
/usr/lib/menues/bin	D-MENU's shell procedurer.
/usr/lib/menues/error	Filer med felinformation.
/usr/USERID/menues	Användarens egna menyer.
/usr/USERID/menues/help	Användarens egna hjälpfiler.

Filer:

/usr/bin/menu	Menyhanteraren och användarinterface. Presenterar menyer och hanterar användarens menyval.
/usr/bin/menuedit	Menyeditor för konstruktion av menyer.
/usr/lib/menues/menutext.def	Innehåller alla texter som D-MENU använder.
/usr/lib/menues/menumsg.def	Innehåller alla feltexter samt viss del av systemmenyernas dialogtexter.
/usr/lib/menues/menucap	D-MENU's terminalberoende sekvenser.
/usr/lib/menues/menuprog	Innehåller terminalinitieringssekvenser för olika program som kan startas upp från menyhanteraren.
/etc/menuaccess.def	Innehåller alla behöriga samt de behörigas behörighetsnycklar.
/etc/menutty.def	Innehåller terminalbehörighet för D-MENU:s användare.
/etc/menuprint.def	Innehåller skrivare som är kopplade till systemets terminaler.
/usr/lib/menues/bin/editor.def	D-MENU's texteditor.
/usr/lib/menues/bin/listfile.def	D-MENU's utmatning av filer på terminal.
/usr/lib/menues/bin/printstand.def	D-MENU's procedur för utskrift på systemets default printer.
/usr/lib/menues/bin/print.def	D-MENU's procedur för utskrift på valbar printer.

OBS! System Administratören behöver ej sätta upp /usr/lib/menues/bin i användarnas PATH variabel. D-MENU sköter detta själv vid uppstart.

7

7. Shell

7.1 Introduktion

7.1.1 Enkla kommandon
7.1.2 Bakgrundskommando - &
7.1.3 Omdirigering av data - > >> <
7.1.4 Pipes och filter - ö
7.1.5 Wildcards i filnamn - * . ? Ä ! Å
7.1.6 Metatecken - ovanstående och ¤ ; ( ) = Ö ` "
7.1.7 Prompt - ¤ > # PS1 PS2
7.1.8 Shell och login, .profile
7.1.9 Sammanfattning

7.2 Shell procedurer

7.2.1 Flödeskontroll - for do done
7.2.2 Flödeskontroll - case esac
7.2.3 Here dokument - << <<-
7.2.4 Shellvariabler - ¤parameter
7.2.5 Testkommando - test
7.2.6 Flödeskontroll - while (until) do done shift
7.2.7 Flödeskontroll - if then elif then else fi
7.2.8 Gruppering av kommandon - (.... ; ....)
7.2.9 Felsökning i shellprocedurer - flags -v -n -x

7.3 Parametertolkning

7.3.1 Parameteröverföring - export readonly
7.3.2 Parameterersättning - ¤äparamå
7.3.3 Kommandoersättning - `....`
7.3.4 Villkor för ersättning - '....' Ö "...."

7.4 Fel som upptäcks av shell

7.4.1 Hantering av fel - trap signals

7.5 Kommandoexekvering

7.5.1 Shell environment - set >& <& background
7.5.2 Anrop av shell - shell options, set options
7.5.2 Begränsad shell - rsh

7.6 Syntax och konventioner

7.6.1 Kommandosyntax
7.6.2 Metatecken och reserverade ord

7.1 Introduktion

Shell är både ett programmeringsspråk och ett kommandospråk men benämns ofta kommandoavkodare till D-NIX.

Shell startas normalt automatiskt när en användare loggar in till systemet enligt den information som finns i filen /etc/passwd. Shell laddas från filen **/bin/sh** och motsvarar den shell som i litteraturen kallas Bourne Shell.

Shell är ett kommandospråk som utgör en koppling mellan användarna och D-NIX operativsystem. I shell finns det möjligheter att styra in- och utdata, tilldela värden till variabler och strängar, och utföra substitutioner. Shell kan arbeta både i interaktiv och icke interaktiv mod.

I kapitel 7.1 beskrivs de detaljer som en 'varje dag' användare behöver känna till, vid läsning av denna del är det en fördel om vissa baskunskaper om D-NIX finns. I kapitel 7.2 beskrivs de grundläggande funktioner som kan användas i shell procedurer. I dessa inkluderas flödeskontrollerade primitiver och värden på strängvariabler som shell tillhandahåller. Vid läsning av detta kapitel är det en fördel om en viss kännedom om programmering finns. I resten av kapitel 7 beskrivs de mer avancerade möjligheterna i shell.

OBS! Detta kapitel (7) innehåller en allmän beskrivning med exempel. För en exakt och fullständig definition av alla shell-kommandon och parametrar hänvisas till beskrivningen av kommandot /bin/sh i D-NIX användarmanual.

7.1.1 Enkla kommandon

I D-NIX skrivs kommandona som en sträng bestående av några få ord särskilda med blanktecken. Första ordet i en sträng är kommandot, resterande ord är argument till kommandot. Kommandona är enkla och korta som t ex

pwd listar ut fullständiga namnet på aktuellt bibliotek

ls -l listar ut alla filnamn i aktuellt bibliotek, -l är en option som anger att inte bara filnamnet skall skrivas ut, utan också information om hur stor filen är, när den modifierades sist, vem som äger den etc.

För de flesta kommando som ges startar shell upp en ny process och väntar tills den har avslutas. Några kommandon är interna i shell men de flesta finns i filer med samma namn som kommandonamnet.

Ett kommando kan ges från tangentbordet när shell har skrivit ut en 'prompt' (normalt ¤). Se kapitel 7.1.7. Det är vanligen möjligt att börja skriva kommandot redan innan prompten kommer eftersom D-NIX sparar en begränsad mängd inmatade tecken från tangentbordet tills något program läser dem (s k "type ahead").

7.1.2 Bakgrundskommando

En del processer kräver en ganska lång exekveringstid. I D-NIX är det därför möjligt att låta dessa exekveras i bakgrunden, kommandot är **&**. Till skillnad mot de övriga kommandona skall detta kommando placeras sist på raden, på följande sätt:

```
cc dts.c &
```

I exemplet kommer programmet dts.c att kompileras i bakgrunden av c-kompilatorn. När processen startas upp kommer shell att skriva ut ett nummer, processnumret, på skärmen. Detta nummer är den enda identifikation som finns till processen, därför är det viktigt att komma ihåg det. Med kommandot **ps** kan en utskrift fås på skärmen över vilka processer som är aktiva just då.

Bakgrundsprocesser kan exekveras med minskad prioritet, genom kommandot **nice**, för att hindra dem att fördröja vanliga förgrundsprocesser.

7.1.3 Omdirigering av data

Shell arbetar alltid mot standard output vilket i normala fall är skärmen. Men i många fall skall utdata behandlas på något sätt, innan den skrivs ut på skärmen. I stora beräkningsprogram kan t ex indata vara utdata från andra beräkningsprogram eller utdata skall sparas på en fil.

I dessa fall kan utdata omdirigeras med tecknet >,t ex.

```
cat > test.doc
```

Shell tolkar endast notationen > test.doc, den skickas alltså inte till kommandot som något argument. I exemplet ovan, kommer utdata att omdirigeras från skärmen till filen test.doc.

```
cat > test.doc
Denna text skrivs in på terminalen som ett exempel på
hur cat fungerar. cat kopierar denna text till en
fil. Avsluta texten med CTRL-D.
```

Filen test.doc kommer att innehålla den inskrivna texten. Om filen inte existerade skapas den innan utdata från **cat** skrivs in. Existerade den däremot kommer det tidigare innehållet i test.doc att raderas och därefter skrivs utdata från **cat** in.

Det är också möjligt att lägga till utdata till en fil som redan existerar, t ex

```
cat >> test.doc
```

Det tidigare innehållet i test.doc behålls och utdata från **cat** läggs till sist i filen.

```
cat > test.doc
Denna text skrivs in på terminalen som ett exempel på
hur cat fungerar. cat kopierar denna text till en
fil. Avsluta texten med CTRL-D.

cat >> test.doc
Detta är ett tillägg till filen test.doc.
Avsluta med CTRL-D.
```

Nu kan **cat** användas för att visa innehållet i filen.

```
cat test. doc

Denna text skrivs in på terminalen som ett exempel på
hur cat fungerar. cat kopierar denna text till en
fil. Avsluta texten med CTRL-D.
Detta är ett tillägg till filen test.doc.
Avsluta med CTRL-D.
```

Det är inte bara utdata som kan omdirigeras utan också indata kan omdirigeras. Normalt tas indata från standard input, tangentbordet, men den kan också tas från någon fil.

```
wc < test.doc

73 342 2755 test.doc
```

Kommandot **wc** räknar antalet rader, ord och tecken i den angivna filen. I exemplet ovan kommer **wc** att ta indata från filen test.doc.

Många kommandon läser indata från filer som ges som kommandoparametrar men läser istället från standard input om ingen infil angivits. Samma sak gäller utdata. För dessa kommandon behövs normalt inte omdirigering med > eller <. T ex kan kommandot ovan även skrivas:

```
wc text.doc
```

Omdirigering kan även göras med andra filnummer ('file descriptors') än standard input och standard output. Ett exempel är standard error till vilken många kommandon sänder eventuella felmeddelanden. Detta är normalt till bildskärmen men kan omdirigeras genom sekvensen `2>file`. Även andra in- och ut-filer kan användas i en shell-procedur genom att kommandot **exec** används.
Exempel:

```
cat  xyzfile  2>errfile  >nyfile
```

I detta exempel kommer eventuella felmeddelanden att skrivas till errfile medan texten från filen xyxfile skrivs till nyfile.

Mer detaljer om omdirigering återfinns under kommandot **sh**.

7.1.4 Pipes och filter

I det föregående avsnittet beskrevs omdirigering av in- och utdata. Förutom detta är det möjligt att dirigera utdata från ett kommando till att bli indata till ett annat kommando, detta kallas för en pipe. Då krävs att det första kommandot verkligen sänder sina ut-data till standard output och nästa kommando läser från standard input. En pipe betecknas med ö (vertikalt streck med internationell teckenuppsättning), generell beteckning är

```
kommando1 arguments ö kommando2 arguments
```

Exempel

```
ls -l ö wc
```

Kommandot **wc** kommer att som indata få utdata från kommandot **ls** -l, dvs **wc** kommer att räkna antalet rader, ord och tecken som finns i utdata från kommandot **ls** -l. De två kommandon sägs bilda en pipeline, effekten är den samma som om följande hade skrivits:

```
ls -l > test.doc
wc < test.doc
```

eller

```
ls -l > test.doc ; wc < test.doc
```

Pipes är enkelriktade och de synkroniseras genom att processen **wc** gör en paus när något skall läsas in om data saknas i pipen, och processen **ls** gör en paus när pipen temporärt är full.

Semikolon (;) separerar kommandon på samma rad ifrån varandra. Den enda skillnaden mellan en pipe och exemplet ovan är att det i en pipe inte behöver skapas en fil för att mellanlagra resultatet från **ls**. I exemplet ovan kommer processerna att exekveras en i taget, dvs först exekveras kommandot **ls** -l och utdata läggas upp på filen test.doc, därefter exekveras kommandot **wc** och indata tas från filen test.doc. När istället pipe används kommer båda processerna att exekveras parallellt.

Ett filter är en viss typ av kommando som läser från standard input, behandlar indata på ett för kommandot specifikt sätt och därefter skriver ut resultatet på standard output. Som exempel på sådana kommandon kan **fgrep** och **sort** nämnas. Filtret **fgrep** väljer från indata de rader som innehåller en specificerad sträng. Se exemplet på nästa sida:

```
ls ö fgrep bak
```

skriver ut de rader som innehåller bak i utdata från **ls**, om det finns några. Det andra exemplet på ett filter är **sort**

```
who ö sort
```

Filtret **sort** sorterar utdata från **who**, vilken består av en lista på de inloggade användarna.

En pipeline kan bestå av mer än två kommandon, t ex

```
who ö grep putte ö sort
```

Denna pipeline skriver ut de rader i **who** som innehåller ordet putte, i alfabetisk ordning.

Man kan även skapa namngivna specialfiler som fungerar som pipes, dvs skrivning sker alltid genom att lägga till data i slutet och läsning raderar alltid lästa data. Men data mellanlagras då alltid på ett skivminne. Se kommandot **mknod**.

7.1.5 Wildcards i filnamn

Shell kan generera filnamn från ofullständiga filnamn som innehåller specialtecken, dessa tecken har en speciell betydelse för shell. När shell på en kommandorad stöter på ett av dessa tecken i något argument, expanderar shell argumentet till en lista av filnamn och sänder denna vidare till kommandot som skall exekveras. Dessa specialtecken brukar kallas för wildcards. Det går att hindra shell att avkoda dessa tecken som wildcards genom kommandot **set** -f eller genom att innesluta tecknen inom apostrofer '....'.

I shell finns följande olika sätt att bilda mönster på.

'*'	Detta tecken matchar alla strängar av tecken, inklusive nullsträngen.
'?'	Detta tecken matchar en ensam bokstav eller tecken.
Ä....Å	Matchar alla tecken som står innanför tecknen Ä Å. Detta är hakparenteser med internationell teckenuppsättning.
Ä.-.Å	Ett par tecken som är åtskilda av - inom Ä Å matchar alla bokstäver emellan, inklusive de angivna bokstäverna.
Ä!...Å Ä!.-.Å	Om första tecknet är ! matchas istället alla tecken utom de som anges inom Ä Å.

OBS! Filnamn som börjar med en punkt (.) matchas aldrig av någon wildcard utan måste alltid anges uttryckligen.

Specialtecken *

* matchar ingen eller flera tecken i ett filnamn, utom första tecknet om detta är en punkt.

```
ls  test*
```

Shell behandlar test* och genererar en lista över de filer i det aktuella biblioteket som börjar på test. Om det inte finns någon fil som matchar test skrivs ett felmeddelande ut. Ett exempel på några filnamn som matchar och några som inte gör det.

test* matchar	test* matchar inte
testning	mtest
test	tes
test12	.test
test.brv	usr.test
testbrev	brevtest
testdir	TEST

Specialtecken ?

Frågetecknet är ett specialtecken som nästan fungerar på samma sätt som * men skillnaden är att ? bara matchar ett enda tecken i ett filnamn, utom första tecknet om detta är en punkt.

```
ls test?
```

Shell genererar en lista över de filnamn som finns i det aktuella biblioteket och som har namn som börjar på test och därefter följs av endast ett enda tecken. Ett exempel på några filnamn som matchar och några som inte gör det.

test?	test?
matchar	matchar inte
testa	testning
test2	test12
testb	test.brv
test8	test.l
	testsl

Specialtecken Ä...Å , Ä.-.Å, Ä!...Å, Ä!.-.Å

En teckengrupp omslutna av Ä Å motsvarar ett tecken i ett filnamn som kan vara vilket som helst av de tecken som är skrivna innanför Ä Å (Ä....Å) eller alla tecken mellan (inklusive) tecknen (Ä.-.Å). Med tecknet ! som första tecken inom Ä Å, inverteras valet genom att alla ASCII-tecken utom de som anges av gruppen tas med. Ä Å är hakparenteser med internationell teckenuppsättning. Shell expanderar ett argument som innehåller en sådan teckengrupp, varje tecken i gruppen substitueras, ett i taget. Listan över de filnamn som matchar skickas av shell till det kommando som skall exekveras. Filnamn som börjar med punkt kan inte anges inom Ä Å utan måste specificeras uttryckligen.

Det första exemplet nedan skriver ut alla filnamn i det aktuella biblioteket som börjar på a e i o u y. I det andra exemplet skrivs de filnamn ut som finns i det aktuella biblioteket som börjar på någon bokstav mellan m och v.

```
ls Äaeiouyå*
.
.
.
ls Äm-vå*
.
.
```

Om filnamnet istället bara består av två bokstäver, kan andra exemplet ovan skrivas på följande sätt:

```
ls Äm-vå?
```

I nedanstående exempel listas alla filer som inte börjar på a eller x.

```
ls Ä!axÅ*
.
.
```

OBS! Om det inte finns några filnamn som passar till mönstret kommer mönstret att förbli oförändrat och skickas som ett argument till kommandot.

Fördelarna med detta sätt att bilda filnamn är att det sparar in på skrivandet, vilket kan vara bra om filnamnen är långa, och att filnamn kan väljas enligt ett bestämt mönster. Dessutom är det möjligt att söka efter filer med hjälp av dessa tecken, t ex

```
echo  /usr/pal/*/dts
```

söker efter och skriver ut alla dts filer som finns i alla bibliotek som finns under /usr/bin/pal.

Det finns ett undantag från de ovan givna reglerna. De filnamn som börjar med tecknet '.' måste matcha exakt, om följande skrivs

```
echo *
```

kommer alla filnamn som inte börjar på '.' att skrivas ut på standard output. För att få de filnamn som börjar med '.' utskrivna måste kommandot skrivas på följande sätt

```
echo .*
```

Detta undantag existerar för att inte en oavsiktlig matchning av namnen skall ske med '.' = aktuellt bibliotek, och '..' = föräldrar-bibliotek.

OBS! Kommandot **ls** undertrycker information angående filerna '.' och '..' om den inledande punkten inte anges uttryckligen.

7.1.6 Metatecken

Alla tecken som har en speciell betydelse för shell, t ex < > * ? & och vissa andra kallas för metatecken. Senare finns en fullständig lista över metatecknen. Om ett tecken föregås av Ö mister detta tecken sin speciella betydelse, om det har någon. Tecknet Ö är 'backslash' dvs bakåtlutande streck i internationell teckenuppsättning.

Första exemplet nedan skriver ut ett frågetecken och andra exemplet skriver ut tecknet Ö.

```
echo Ö?

echo ÖÖ
```

Betydelsen av 'nyrad'-tecknet (return/line-feed) kan tas bort vid inmatning av en textrad varvid det är möjligt att skriva in t ex ett kommando med parametrar längre än 80 tecken, trots att texten visas på flera rader på bildskärmen. Detta görs genom att skriva Ö som sista tecken på 'raden', varvid nyrad-tecknet ignoreras av shell vid avkodning av raden som ett kommando.

Detta sätt att skriva är tillämpbart då det bara gäller att ta bort specialbetydelsen för ett eller två tecken itaget, men om det ska göras för flera tecken i en sträng blir den beskrivna metoden både klumpig och tidskrävande. Istället kan apostrofer ' ' användas, denna metod är den enklaste och mest användbara. Sätt tecknen ' ' på varsin sida av strängen, t ex

```
echo xx'******'xx
```

Strängen xx******xx skrivs ut på standard output. Denna sträng får innehålla andra metatecken utom apostrof ' som är undantagen från denna regel. Tecknet Ö har dessutom ofta speciell betydelse för olika kommandon, bl a för **echo**, varför det ska användas varsamt även inom ' '.

Det finns också en tredje metod, citationstecknen " " sätts på varsin sida om strängen. Men dessa tecken kan inte ta hand om alla metatecken. Se kommandot **sh** i **AH** för närmare detaljer.

7.1.7 Prompt

När en terminal används som standard input kommer shell att skriva ut en prompt innan den kan ta emot ett kommando. Shell ger som default, standardvärde, prompten ¤. Denna kan ändras genom att variabeln PS1 tilldelas ett nyttvärde. I nedanstående exempel visas prompten såsom den syns på bildskärmen. T ex

```
¤ date
Fri May 03 10:43:13 PST 1985
¤ PS1='hej på dej!'
hej på dej! date
Fri May 03 10:44:23 PST 1985
hej på dej!
.
.
.
hej på dej!
```

på skärmen kommer hej på dej! att skrivas ut istället för ¤. Prompten # indikerar att användaren för tillfället befinner sig i superuser mod.

7.1.8 Shell och login

Innan användaren loggar in kan ett allmänt systemmeddelande visas (ur filen /etc/issue), eventuellt följt av ett systemnamn. Efter inloggningen visas ytterligare ett meddelande (aktuellt för dagen) ur filen /etc/motd.

Därefter startas normalt en shell för att läsa och exekvera de kommandon som skrivs på standard input, normalt terminalen. Om användarens bibliotek innehåller filen .profile, kommer shell att först läsa .profile eftersom denna fil förmodas innehålla diverse kommandon som är av betydelse i detta läge t ex bildskärmstyp, specialtangenter på tangentbordet. Först därefter kommer shell att gå till standard input och läsa kommandon.

7.1.9 Sammanfattning

* Orden i en kommandosträng särskiljs med blanktecken.

* & är kommandot för att en process skall exekveras i bakgrunden. Detta tecken skrivs då sist på raden.

* `ls > fil`

  Utdata från ls omdirigeras från standard output till att skrivas på fil.

* `ls >> fil`

  Utdata från ls läggas till fil, utdata kommer att läggas till i slutet av filen dvs efter det tidigare innehållet i filen.

* `wc < fil`

  Kommandot **wc** får indata från filen fil istället för ifrån standard input.

* `ls ö wc -l`

  Tecknet ö är en pipe, vilket innebär att utdata från ls är indata till **wc**. Resultatet av kommandoraden skrivs på standard output och innehåller information om antalet rader som ingår i aktuellt bibliotek.

* `ls ö fgrep old`

  Resultatet av kommandoraden skrivs ut på standard output och innehåller information om vilka filnamn som innehåller strängen old.

* `ls ö fgrep old ö wc -l`

  Resultatet från kommandoraden skrivs ut på standard output och innehåller information om hur många filer i detta aktuella bibliotek som strängen old ingår i.

* `cc pgm.c &`

  Kompilerar programmet pgm.c i bakgrunden.

* Wildcards

*	Matchar alla strängar inklusive nollsträngen.
?	Matchar ett tecken, vilket som helst.
Ä...Å	Matchar alla inneslutna tecken.
Ä.-.Å	Matchar alla tecken mellan och inklusive de angivna.
Ä!...Å	Matchar alla tecken utom de inneslutna.
Ä!.-.Å	Matchar alla tecken utom de som matchas av Ä.-.Å.

OBS! En punkt som första tecken i ett filnamn matchas aldrig av en wildcard.

7.2 Shell procedurer

Shell kan inte bara exekvera enkla kommandon utan också läsa och exekvera kommandosträngar som ligger i en fil. När kommandot **sh** används behöver användaren inte ha exekveringstillstånd till filen. Exempel:

```
sh fil argument
```

sh startar en ny shell och anger att kommandon skall läsas från filen fil. Denna fil kallas för kommandoprocedur eller shellprocedur. Argumenten kan inkluderas i anropet, i dessa fall refererar argumenten till de sk positions parametrar, ¤1, ¤2,.., i filen, t ex antag att filen drs innehåller proceduren

```
who ö fgrep ¤1
```

skriv på terminalen

```
sh drs putte
```

I kommandot i filen drs ersätts ¤1 med första positionsparametern som i detta fall är putte. Exemplet ovan är likvärdigt med

```
who ö fgrep putte
```

En fil kan i sig själv vara exekverbar om 'rätt' åtkomstprivilegier anges. I D-NIX finns det tre olika privilegier: läsbar, skrivbar och exekverbar. Varje fil kan ha en, två eller alla tre av dessa privilegier angivna samtidigt. Dessutom kan olika privilegier anges för ägaren, gruppen och speciellt för den grupp som kallas 'others'. Privilegier som en fil har kan ändras med kommandot **chmod**. I exemplen nedan ändras privilegier för alla.

Om privilegiet för filen drs ändras till att vara exekverbar

```
chmod +x drs
```

kommer uttrycket

```
drs putte
```

att vara likvärdigt med

```
sh drs putte
```

Denna funktion tillåter att program- och shellprocedurer används blandat. I båda fallen kommer en ny process att skapas för varje nytt kommando som skall exekveras. Denna process avslutas normalt då kommandot eller shellproceduren är avslutad. Jämför kommandot **exec**, vilket istället gör att nuvarande shell-process byts ut mot en ny.

7.2.1 Flödeskontroll - for do done

En shellprocedur används ofta för att genomlöpa en loop av argument (¤1,¤2,¤3,...) där de angivna kommandona exekveras för var och en av argumenten. Ett exempel på en sådan procedur är proceduren telefon som genomsöker filen /usr/lib/telnr, filen innehåller namn och telefonnummer. **cat** används nedan för att se filens innehåll.

```
cat /usr/lib/telnr
.
.
.....................
Eva Karlsson 675787
Putte Engby  124848
.....................
.
```

Texten i proceduren telefon är följande.

```
cat telefon
for i
    do
    fgrep -i "¤i" /usr/lib/telnr
    done
```

Om filen telefon har exekveringsprivilegier satta kan telefonnummer visas som i exemplet nedan.

```
telefon putte
Putte Engby   124848
```

Kommandot telefon skriver ut alla rader i filen /usr/lib/telnr som innehåller ordet putte. Optionen -i till **fgrep** anger att stora och små bokstäver behandlas lika.

Den generella formen för en for-loop är:

```
for name in w1 w2 w3 w4 ......
    do kommandolista
    done
```

name måste bestå av en shellvariabel som sätts till respektive w1 w2 w3 w4 .... varje gång loppen med kommandolista och do exekveras. Om 'in w1 w2 w3 w4 .....' utesluts kommer loopen att exekveras en gång för varje positionsparameter, dvs shell antar att 'in w1 w2 w3.....' är 'in ¤1 ¤2 ¤3 ....' Kommandolista kan innehålla en eller flera kommandon separerade eller avslutade antingen med semikolon eller med newline. Orden **do** och **done** är reserverade ord och känns endast igen efter semikolon eller på en ny rad.

Ytterligare ett exempel på hur **for** kan användas är kommandot create,

```
cat create
for i
    do > ¤i ; done
```

Prova kommandot med:

```
create fil1 fil2
```

Kommandot create skapar för varje angivet argument, i det här fallet fil1 och fil2, två tomma filer. Om de fanns tidigare försvinner nu eventuellt gammalt innehåll. OBS! Om done står på samma rad som kommandot måste ett semikolon föregå done.

7.2.2 Flödeskontroll - case esac

Case är en villkorssats som tar hand om de fall när flera olika händelser inträffar, exemplet nedan visar ett kommando som vi kallar append.

```
cat append

case ¤#  in
        1) cat >> ¤1 ;;
        2) cat >> ¤2 < ¤1 ;;
        *) echo 'syntax: append (från) till' ;;
esac
```

Om kommandot append endast skrivs med ett argument, som t ex

```
append test.doc
```

inträffar punkt 1) i kommandot append , dvs positionsparametern ¤# kommer att ersättas med strängen 1, vilket medför att text från standard input kommer att läggas till sist i filen test.doc. I detta fall läses standard input från tangentbordet, och avslutas med tangenten CTRL-D. Om kommandot skrivs med två argument som t ex

```
append test.doc read.brv
```

inträffar punkt 2) i kommandot append , vilket innebär att innehållet i test.doc läggs till sist i filen read.brv. Om inget eller fler än två argument anges skrivs raden *) ut.

Den generella syntaxen för case är:

```
case ord in
     mönster 1) kommandolista ;;
     mönster 2) kommandolista ;;
     mönster 3) kommandolista ;;
     .
     .
esac
```

Varje kommandolista måste avslutas med två semikolon (;;).

Shell försöker att jämföra värdet av shellvariabeln ord med varje mönster, i den ordning som mönster förekommer. Om shell finner ett mönster som motsvarar ord, kommer kommandolistan att exekveras. OBS! Det finns ingen kontroll på att bara ett mönster passar, utan det första mönster som **case** hittar kommer att exekveras. I exemplet nedan kommer aldrig kommandona efter den andra *) att utföras.

```
case ¤# in
        *) ... ;;
        *) ... ;;
esac
```

Ett annat användningsområde för case är att urskilja olika former av argument. Nedanstående exempel är ett fragment av ett cc kommando.

```
for i ; do case ¤i in
                -ÄocsÅ) ... ;;
                    -*) echo 'okänd flagga ¤i' ;;
                   *.c) /lib/c0 ¤i ........ ;;
                     *) echo 'oväntat argument ¤i' ;;
esac ; done
```

Om ett kommando passar in på flera olika mönster, använd tecknet ö till att separera de olika mönstren. Ä...Å kan även användas. Kommandot case kan antingen skrivas som

```
case ¤i in
     -x ö -y) ..... ;;
esac
```

eller som

```
case ¤i in
     -ÄxyÅ)  ...... ;;
esac
```

De notationskonventioner för metatecken som nämnts tidigare gäller även för kommandot case:

```
case ¤i in
     Ö?)   ....
esac
```

matchar tecknet ?.

7.2.3 Here dokument - << <<-

Tidigare använde vi filen telnr till att förse kommandot **fgrep** med data. Istället för att data ligger i en separat fil kan data innefattas i shellproceduren, detta kallas för ett 'here dokument', t ex

```
cat telefon

  for i
    do fgrep "¤i" <<+
    ...
    Eva Karlsson
    Putte Engby
    ...
    +
    done
```

Shell kommer i ovanstående exempel att ta all text som står mellan << + och + som standard input för **fgrep**. Notera att + måste stå i början av en rad. Istället för + kan ett valfritt ord användas för att markera början och slutet av here-dokumentet.

Innan parametrarna i dokumentet är åtkomliga för **fgrep** kommer de att substitueras, vilket visas i nedanstående exempel. Detta kan förhindras genom att låta något tecken i begränsningsordet föregås av Ö.

Exempel på substituering i ett kommando i filen edg:

```
cat edg

ed ¤3 <<+
g/¤1/s//¤2/g
w
+
```

Om innehållet i den exekverbara strängen edg är enligt ovanstående så är kommandot

```
edg stringl string2 fil
```

likvärdigt med kommandot

```
ed fil <<+
g/stringl/s//string2/g
w
+
```

som ändrar alla förekomster av string1 i fil till string2.

7.2.4 Shellvariabler

I shell kan strängvariabler tilldelas ett värde. Namnen på strängvariablerna skall alltid börja med en bokstav, resterande tecken kan bestå av bokstäver, siffror och understrykningstecken. En variabel kan tilldelas ett värde på tre olika sätt, se exempel

```
anv=kurt   csmn=m000   dir=mh000
```

Variablerna anv, csmn och dir tilldelas resp värde. En variabel kan sättas till nollsträngen på följande sätt. Notera att variabeln fortfarande är definierad i shell.

```
null=
```

För att helt ta bort en variabeldefinition används kommandot **unset**.

```
unset csmn
```

När en variabel skall substitueras med sitt värde skall tecknet ¤ anges före variabelnamnet, på följande sätt

```
echo ¤anv
```

Variabeln anv kommer att substitueras mot värdet av denna variabel dvs kurt och kommandot **echo** visar värdet på bildskärmen.

För att underlätta skrivandet kan förkortningar användas för strängar som förekommer ofta, t ex

```
b=/usr/kalle/bin
mv pgm ¤b
```

Kommandot mv kommer att flytta filen pgm till biblioteket ¤b dvs /usr/kalle/bin.

En generell beteckning för variabelsubstitution är

```
echo ¤äanvå
```

Tecknen ä å motsvarar klammertecken med internationell teckenuppsättning. I detta enkla fall är ovanstående likvärdigt med:

```
echo ¤anv
```

Beteckningen med ä å måste användas när ett variabelnamn följs direkt av en bokstav eller en siffra, vilken annars skulle tagits som en del av variabelnamnet. Till exempel

```
tmp=/tmp/ps
ps -a > ¤ätmpåfil
```

Utdata från **ps** kommer att skrivas till filen /tmp/psfil.

Om det istället hade stått

```
ps -a > ¤tmpfil
```

hade varibeln tmpfil substituerats med sitt värde.

Alla variabler som räknas upp här nedan sätts då shell startas utom ¤?, denna parameter sätts först efter exekvering av ett kommando.

¤* Hela parametersträngen med alla parametrar utom ¤0

¤0 Kommandot själv

¤1 etc Positions-parametrarna i kommandosträngen. Jämför shell-kommandot **shift** som skiftar innehållet i positionsparametrarna ett eller flera steg. **shift** beskrivs tillsammans med kommandot **while**.

¤# Ger antalet positionsparametrar i decimal form. Se t ex exemplet 'append' för shell-kommandot **case**.

¤É Samma som ¤*, men behandlas annorlunda inom citationstecken "...".

¤? Det sist exekverade kommandots utgångsstatus (returnerad kod), i decimal form. Den returnerade utgångsstatusen från de flesta kommandon är om kommandot lyckas noll och om det misslyckas icke-noll. Se vidare shell-kommandona **if** och **while**.

¤¤ Anger processnumret för det shell som är aktuellt, i decimal form. Eftersom varje process har sitt unika nummer används ¤¤ ofta till att skapa temporära filer, exempelvis

```
ps -a > /tmp/ps¤¤
...
rm /tmp/ps¤¤
```

¤! Anger processnumret på den sista process som exekverades i bakgrunden, i decimal form.

¤- Visar de shell-optioner som gäller vid det aktuella tillfället. Dessa kan vara satta då shell startades eller med kommandot **set**.

En del variabler har en speciell betydelse som endast används vid speciella tillfällen.

¤HOME Defaultargumentet för kommandot cd. Ett filnamn som anges utan /, antas ligga i det biblioteket som är det aktuella biblioteket. Det aktuella biblioteket ändras med hjälp av cd kommandot. Till exempel

```
cd /usr/bin/kurt
```

ändrar det aktuella biblioteket till /usr/bin/kurt. Kommandot **cd** kan anges utan argument, vilket är ekvivalent med att skriva

```
¤ cd ¤HOME
```

Variabeln ¤HOME sätts automatiskt vid inloggning, men kan ändras vid behov.

¤PATH Innehåller en lista på de bibliotek som innehåller kommandon. Varje gång ett kommando exekveras av shell utan att ett fullständigt kommandofilnamn som börjar med / anges, söks denna lista igenom efter en exekverbar fil. Om ¤PATH inte är satt kommer som default följande bibliotek att genomsökas i följande ordning: aktuella biblioteket, /bin och /usr/bin. I ¤PATH skrivs biblioteken in åtskilda av : (kolon), exempel

```
PATH=:/usr/kurt/bin:/bin:/usr/bin
```

Strängen ovan specificerar att det aktuella biblioteket (nollsträngen innan första :), /usr/kurt/bin, /bin och /usr/bin ska genomsökas i denna ordning. Detta medför att användaren kan komma åt sina egna kommandon oberoende av det aktuella biblioteket. Om namnet på kommandot innehåller / kommer inte den just beskrivna sökningen av biblioteken att utföras, ett enda försök kommer att göras att exekvera kommandot.

¤CDPATH

Innehåller en lista på de bibliotek som ska avsökas då kommandot cd exekveras för att leta efter det bibliotek som anges om biblioteket anges utan inledande /. Formen är likadan, med kolon, som för variabeln ¤PATH. Om ¤CDPATH inte är definierad söks bara i det aktuella biblioteket.

¤PS1 Promptsträngen i första shell, default ¤.

¤PS2 Shell prompt när mer indata behövs, default '>'.

¤IFS En sträng med alla tecken som shell anser som fältavgränsare mellan ord. Se kapitel 7.3.4.

¤MAIL Denna variabel anger den fil dit kommandot mail sänder meddelanden till användaren. När shell startas i interaktiv mod, t ex vid inlogging eller då sh-kommandot ges utan argument, och med jämna mellanrum därutöver enligt variabeln ¤MAILCHECK kontrolleras den specificerade filen innan promten skrivs ut. Om filen har ändrats sedan sist, skrivs följande ut 'You have mail', därefter är shell redo att ta emot ett kommando. Variabeln ¤MAIL sätts automatiskt vid inlogging enligt användarnamnet enligt exemplet nedan (namnet är kurt).

```
MAIL=/usr/spool/mail/kurt
```

¤MAILCHECK
Värdet anger tiden i sekunder mellan kontrollerna av filen ¤MAIL för att informera användaren om att meddelande har kommit.

¤MAILPATH
Detta är en kolon-separerad lista med meddelande-filnamn. Här kan till skillnad från i ¤MAIL flera filer specificeras och dessutom kan textsträngar anges vilka printas istället för texten "You have mail" då ett meddelande kommer. Olika texter kan anges för olika filer genom att filnamnet följs av tecknet % och motsvarande textsträng.

7.2.5 Testkommando

Kommandot **test** returnerar status noll om ett angivet villkor är uppfyllt och icke-noll om det inte är uppfyllt. Villkoret kan t ex vara om en fil existerar eller ej såsom i följande exempel:

```
if test -f fil
   then echo "Filen existerar"
   else echo "Filen finns inte"
fi
```

Generellt beräknar **test** ett värde som returneras som utgångsstatus för kommandot. Vanligen görs tester på shellvariabler. Några av de mest frekventa testargumenten återges här nedan.

```
test  "s"    sant om argumentet s inte är nollsträngen
             I detta fall bör argumentet inneslutas av
             " " eftersom test ger ett felmeddelande om
             argumentet helt saknas. Citationstecknen
             krävs för att test ska kunna skilja på ett
             null-argument och om argumentet saknas.

test -f fil sant om fil existerar som vanlig fil.
test -r fil sant om fil existerar och är läsbar
test -w fil sant om fil existerar och är skrivbar.
test -d fil sant om fil existerar och är ett
            bibliotek.

test n1 -eq n2  sant om n1 och n2 är heltal och lika.
```

Ett annat exempel är en del ur en shell-procedur där kommandot testar om ett filnamn slutar med .bak'

```
if test ¤1 != `basename ¤1 .bak`
   then echo "Filnamnet slutar med .bak"
   else echo "Filnamnet slutar inte med .bak"
fi
```

En fullständig beskrivning av **test** finns i **AH**.

7.2.6 Flödeskontroll - while (until) do done, shift

I shell finns det en **while** loop, en **until** loop och en **if then else fi** sats, hur dessa fungerar beror på vilket utgångsstatus de kommandon har som skall behandlas. Den generella formen för while loopen är:

```
while  kommandolista1
do     kommandolista2
done
```

while testar på värdet på utgångsstatusen för det sista enkla kommandot i kommandolista1, som exekveras varje gång som loopen genomlöps medan kommandolista2 bara exekveras om utgångsstatus lika med noll har returnerats från kommandolista1. Skulle utgångsstatusen vara skild från noll avslutas **while**-kommandot. Exempel:

```
while test ¤1
  do ........
     shift
  done
```

är likvärdigt med

```
for i
  do .......
  done
```

Shift är ett shellkommando som ändrar namnen på positionsparametrarna ¤2, ¤3, ... till ¤1, ¤2, ..., vilket medför att ¤1 försvinner.

Ytterligare en användning av while eller until loopen är att den väntar på att en extern händelse skall inträffa, först därefter exekveras den specificerade kommandolistan.

En **until** loop avslutas på ett annat sätt än while loop, t ex enligt följande exempel i filen testf.

```
cat testf

until test -f fill
      do sleep 300
      done
kommandon
```

Loopen kommer att var 5:e min testa om filen fil1 har skapats och först därefter exekveras kommandona efter **until**-loopen. (fil1 kommer troligtvis att skapas av någon annan process).

Det är möjligt att bryta sig ur en **while**-, **until**- eller **for**-loop även utan att villkoret är uppfyllt genom kommandot **break**. På liknande sätt kan nästa varv i en **while**- eller **for**-loop påbörjas utan att resten av kommandona i kommandolistan genomlöps genom kommandot **continue**.

7.2.7 Flödeskontroll - if then elif else fi

I shell finns det också en generell villkorssats, **if**.

```
if      kommandolista1
then    kommandolista2
else    kommandolista3
fi
```

if testar utgångsstatus från på det sista enkla kommandot i kommandolista1. Om detta är noll exekveras kommandolista2. Om det är skilt från noll exekveras istället kommandolista3.

if kan användas ihop med kommandot **test** för kontroll av om en fil existerar.

```
if test -f fil2
then    behandla fil2
else    gör något annat
fi
```

Existerar fil2 kommer den för filen specificerade processen att utföras, i annat fall kommer det som står efter else att utföras.

Ett exempel på hur **if**, **case** och **for** kan användas ges senare.

En multipeltest av **if** kan se ut på följande sätt:

```
if ...
   then ...
   else if ...
           then ...
           else if ...
                   ...
                fi
        fi
fi
```

För att undvika denna ganska klumpiga konstruktion kan en utvidgning av **if** användas.

```
if ...
then ...
elif ...
then ...
elif ...
...
fi
```

Som ett exempel på detta kan kommandot **mytouch** skapas. **mytouch** kan användas till att ändra sista modifieringstiden för i-noden för en eller flera filer. Denna tid ses med följande kommando:

```
l -c  filnamn
```

Kommandot ligger i filen mytouch som kan ses med **cat**:

```
cat mytouch

flag=
trap "rm -f junk¤¤ ; exit" 1 2 3 15
for i
  do case ¤i in
      -c) flag=N ;;
       *) if    test -f ¤i
          then  ln ¤i junk¤¤ ; rm junk¤¤
          elif  test ¤flag
          then  > ¤i
          else  echo fil Ö'¤iÖ' existerar ej
          fi ;;
     esac
  done
```

I detta exempel används flaggan -c till att tvinga filer att skapas om de inte redan existerar. Ett felmeddelande skrivs annars ut om filen inte existerar. Om argumentet -c hittas kommer shell att sätta shellvariabeln flag till en icke-noll sträng. Kommandona

```
ln ...
rm ..
```

länkar sig till filen och tar därefter bort den, detta medför att i-nodens modifieringstid uppdateras för filen. Däremot inte den modifieringstid som gäller för filinnehållet, det som ses med kommandot **l** utan optionen -c.

Sekvensen

```
if kommando1
then kommando2
fi
```

kan skrivas enklare

```
kommando1&&kommando2
```

Jämför detta med

```
kommando1 öö kommando2
```

Kommando2 exekveras i detta fall (med öö) bara om exekveringen av kommando1 har misslyckats, dvs om utgångsstatus från kommando1 är skilt från noll.

7.2.8 Gruppering av kommandon

En grupperingen av kommandona kan ske på två olika sätt, antingen med

```
kommando-lista;
```

eller

```
(kommando-lista)
```

I det första fallet exekveras bara kommandolistan, medan den exekveras som en separat process i det andra fallet. Exempel:

```
(cd xdir ; rm junk)
```

rm junk exekveras i biblioteket xdir utan att det aktuella biblioteket ändras i det shell som anropats. Kommandolistan exekveras nämligen av en ny shell som avslutas efter **rm**-kommandot. Grupperingen

```
cd xdir ; rm junk
```

utför samma sak som i exemplet ovan men det shell som anropats lämnas kvar i biblioteket xdir.

Det går även att ge namn åt en kommando-lista. Därvid definieras en namngiven funktion inom shell. Denna funktion kan sedan användas på samma sätt som vanliga kommandon, med den skillnaden att kommandolistan exekveras inom nuvarande shell istället för i en ny process. Argument kan ges, vilka blir temporära positionsparametrar ¤1, ¤2, ... Funktioner definieras som i följande exempel, där det är väsentligt att ä å omger listan och att listan avslutas med ; eller ny-rad. ä å motsvarar klammertecken med internationell teckenuppsättning.

namn1() ä kommandolista ; å

7.2.9 Felsökning i shellprocedurer

I shell finns det två olika verktyg för debuggning. Det första anropas i en procedur med

```
set -v
```

(v = mångordig). Shell kommer att skriva ut kommandoraderna exakt som de läses innan varje rad exekveras. Med detta verktyg kan isolerade syntax fel letas upp. En procedur kan anropas utan att procedurens funktion i övrigt påverkas:

```
set -vn kommando parametrar
```

Som kommando skall en shellprocedur anges.

Optionen -v kan användas tillsammans med optionen -n för att förhindra exekvering av de kommandon som kommer efter set. Då listas enbart de lästa raderna ur shellproceduren.

OBS! Om set -n skrivs på en terminal förblir denna obrukbar tills ett end-of-file ges.

Det andra sättet att utföra en debuggning på är göra 'trace' under exekveringen:

```
set -x
```

Kommandona kommer att skrivas ut allt eftersom de exekveras. Kommandot **set** kan användas utan optioner, varvid **set** listar alla definierade shellvariabler, shell- och set-optioner och positionsparametrar.

```
set
```

De optioner som är satta vid det aktuella tillfället kan fås fram med

```
echo ¤-
```

Dessa och flera andra optioner beskrivs under kommandot **set** i **AM**.

7.3 Parametertolkning

En shellvariabel kan tilldelas ett värde vid en variabelsubstitution eller tilldelas ett värde innan en shellprocess anropas. Variabelns värde kan ändras med namn=värde, men då ändras variabelns värde enbart i den shell som exekverar. Enbart om variabeln exporteras (med kommandot **export** eller genom att kommandot **set** -a givits) kommer dess nya värde att vara definierat i anropade kommandon. Alternativt kan värdet ändras i ett anropat kommando genom att namn=värde anges i kommandoraden före kommandot. Då ändras variabelvärdet bara i den nya process som startas upp och inte i den shell, varifrån kommandot ges.

```
anv=kurt kommando
```

Variabeln anv kommer att ha värdet kurt i det kommando som exekveras. Om kommandot **set** -k har givits kan argument av typ namn=värde införas var som helst i argumentlistan. Dessa shellvariabler kallas ibland för tangentbordsparametrar. De återstående argumenten, om det finns några, är tillgängliga som positionsparametrarna ¤1,¤2,... .

Med kommandot **set** kan positionsparametrarnas värde ändras inne i en process, exempel:

```
set -- *
```

I detta exempel används en 'wildcard' (*) som utvecklas till alla filnamn i det aktuella biblioteket. Positionsparametern ¤1 kommer att tilldelas det första filnamnet i det aktuella biblioteket, ¤2 det andra osv. OBS! Det första argumentet, --, försäkrar korrekt behandling även av filnamn som börjar på -, eftersom det hindrar **set**-kommandot från att tolka eventuella strängar som börjar på - som optioner. Jämför kommandot **set** i **AM**.

7.3.1 Parameteröverföring

I ett anrop av en shell procedur kan både positions- och tangentbords-parametrar anges i kommandoraden. Shellparametrar i övrigt kan göras tillgängliga för en anropad process genom att det i förväg specificeras att dessa parametrar skall exporteras, exempelvis:

```
export anv ccs
```

Variablerna anv och ccs är märkta för export.

Vid anrop av en shell process kommer alla exporterade variabler att kopieras och kan därvid modifiera den shell som anropas. Allmänt gäller det att variabler som ändras inom en process aldrig kan återföras till den anropande shell-processen. En process kan aldrig modifiera den shell som startat den nya processen.

Med kommandot **set** -a anges att alla shellvariabler som skapas eller ändras efter detta kommando, automatiskt markeras för export.

Om variabler deklareras endast för läsning, read-only, kan den anropade processen hindras från att ändra dessa variabler. Detta kommando har samma syntax som export.

```
readonly namn ...
```

Obs! Detta kommando måste, liksom export, specificeras innan den nya shell processen anropas.

7.3.2 Parameterersättning

Om en shellparameter inte är satt, ersätts (substitueras) den med null-strängen, t ex om variabeln ddd inte har tilldelats något värde kommer någon av följande kommandon inte att skriva ut något alls, utom en radframmatning.

```
echo ¤ddd

echo ¤ädddå
```

En defaultsträng kan väljas istället för 'ingenting' om en variabel är odefinierad, om följande format används.

```
echo ¤äddd-TOMå
```

Detta kommando skriver ut värdet på ddd om den har något, i annat fall skrivs strängen 'TOM' ut. I defaultsträngen är de vanliga konventionerna tillåtna,

```
echo ¤äddd-'*'å
```

skriver ut * om variabeln ddd inte har något värde. På samma sätt kommer

```
echo ¤äddd-¤1å
```

att skriva ut ddd om den har något värde, i annat fall skrivs värdet på positionsparametern ¤1 ut, om den är satt till något värde. En variabel kan tilldelas ett defaultvärde, om den inte har något värde:

```
echo ¤äddd=.å
```

vilket ger samma resultat som:

```
if test "¤ddd"
   then echo ¤ddd
   else ddd=. ; echo ¤ddd
fi
```

Med denna variabelsubstitution sätts ddd till . (punkt) om variabeln inte är satt sedan tidigare. Positionsparametrarna kan däremot inte ändras på detta sätt.

Med nedanstående exempel:

```
echo ¤äddd?messageå
```

skrivs värdet på variabeln ddd ut om den har något värde, i annat fall kommer shell att skriva ut texten 'message' och exekveringen av proceduren avbryts. Om ingen text anges efter frågetecknet, kommer ett standard-meddelande att skrivas ut: 'parameter null or not set'.

En shell procedur som måste ha en del startparametrar satta, kan inledas av test-kommandon, t ex enligt följande:

```
: ¤äuser?saknaså  ¤äacct?saknaså ¤äbin?saknaså
........
```

: är ett kommando i shell och det gör ingenting när väl dess argument har tilldelas värden. Om någon av variablerna user, acct eller bin inte är satta kommer shell att skriva ut textsträngen för den första som inte definierats och överge proceduren.

Ytterligare varianter på olika sätt att ersätta shellvariabler med värden beskrivs i kapitlet om **sh** i **AM**.

7.3.3 Kommandoersättning

Standard output med data från ett kommando kan substitueras på samma sätt som en parameter. Hela strängen som står mellan tecknen `....` tar shell som ett kommando som skall exekveras, varefter strängen ersätts med utdata från kommandot. Exempelvis listar kommandot **pwd** på standard output det fullständiga namnet på det aktuella biblioteket. Om standard output för detta kommando är /usr/kurt/bin så är kommandot

```
ddd=`pwd`
```

likvärdigt med

```
ddd=/usr/kurt/bin
```

Ett annat exempel är:

```
ls `echo "¤1"`
```

som är ekvivalent med

```
ls ¤1
```

Kommando substitution kan användas överallt där parametersubstitution kan användas, inklusive i here-dokument och behandlingen av resultaten är i båda fallen den samma. Detta medför att kommandon kan användas i shell procedurer för att ändra strängar. Ett exempel är kommandot **basename** som tar bort suffixet från en sträng.

```
basename main.c .c
```

Kommandot **basename** skriver ut strängen main genom att ändelsen '.c' tas bort. **basename** tar bort eventuellt inledande del biblioteksnamn fram till sista snedstrecket (/) (se kommandot **basename**). Hur detta kan användas för substitution visas i följande lilla utdrag ur en shell-procedur. Variabeln A antas vara ett filnamn, från vilket suffixet .c skall tas bort. Variabeln B erhåller filnamnet utan suffix.

```
case ¤A in
    ...
    *.c)    B=`basename ¤A .c`
    ...
esac
```

Andra exempel visas nedan:

for i in `**ls** -t` ; **do** ..... ; **done**

Variabeln i sätts till filnamnen i tidsordning, den senast ändrade används först, enligt -t optionen för kommandot **ls**.

set `**date**` ; **echo** ¤3 ¤2 ¤6, Klockan ¤4

Skriver ut t ex: 20 Mar 1987, Klockan 13:59:59
genom att kommanot **set** sätter positionsparametrarna till de strängar som kommandot **date** ger.

7.3.4 Villkor för ersättning - '....' Ö "...."

Shell är en makroprocess som utför parametersubstitution, kommando-substitution och skapar filnamn med hjälp av wildcards, som argument till kommandona. I detta kapitel kommer det att diskuteras i vilken ordning dessa beräkningar utförs och vilken effekt olika beteckningsmekanismer har.

Initialt analyseras kommandona enligt de grammatiska regler som listas kortfattat i kapitel 7.6. Innan ett kommando exekveras utförs dessutom följande substitutioner i kommandosträngen, varvid olika uttryck ersätts av motsvarande strängar.

* Parameter-substitution, t ex ¤user.

* Kommando-substitution, t ex `pwd`.

  I dessa fall kan endast en beräkning göras, om inte kommandot **eval** används. T ex om värdet på variabeln x är ¤y kommer följande att skriva ut ¤y på bildskärmen och inte värdet av variabeln y. Notera då att apostrofer måste användas vid definitionen av x för att ¤-tecknet inte ska tolkas som en substitution vid definitionen av x.

  ```
  x='¤y'
  echo ¤x
  ```

  kommer att skriva ut ¤y.

* Tolkning av blanktecken.

  Resultaten från de ovan gjorda substitutionerna delas in i ord som inte innehåller några blanktecken (blank interpretation). De tecken som anses vara blanktecken är definierade i shellvariabeln ¤IFS. Åtminstone mellanslag och tab-tecken och ny-rad ingår där. Nollsträngen betraktas inte som ett ord om den inte är citerad, t ex

  ```
  echo ''          (två stycken spostrofer)
  ```

  kommer att som första argument ge nollsträngen till kommandot **echo** medan

  ```
  echo ¤null
  ```

inte kommer att tilldela **echo** något argument alls, om variabeln null inte är satt eller innehåller nollsträngen. Om man vill vara säker på att ¤null ska uppfattas som nollsträngen kan "...." användas, vilket inte hindrar att substitutionen utförs.

```
echo "¤null"
```

* Skapande av filnamn med hjälp av wildcards.

Varje ord genomsöks efter tecknen *, ? och Ä...Å och en alfabetisk lista skapas över filnamnen för att ersätta orden. Var och ett av dessa filnamn blir ett separat argument.

Den just beskrivna ersättningarna förekommer också på den lista av ord som tillhör en **for**-loop. I **case**-satserna kan endast de två första typerna av substitutionerna utföras, och wildcards är lokala inom **case**-satsen.

För att hindra denna ersättning av ett uttryck med sitt motsvarande sträng-värde kan, förutom de tidigare beskrivna metoderna Ö och '.....', även en tredje mekanism användas, "..... " (citationstecken). Till skillnad mot de två övriga kan fortfarande parameter-och kommandosubstitution göras, men den förhindrar att filnamn skapas med hjälp av wildcards och tolkning av blanktecken. Följande tecken har en speciell betydelse innanför "....." och måste föregås av Ö.

```
¤  parameter-substitution
`  kommando-substitution
"  avslutar en citeringssträng
Ö  citerar speciella tecken ¤ ' " Ö
```

Exempel:

```
echo "¤x"
```

ger värdet av variabeln x till **echo**. På samma sätt

```
echo "¤*"
```

ger positionsparametrarna som argument till **echo**. Detta är ekvivalent med

```
echo "¤1 ¤2 ..."
```

Beteckningen ¤É har samma betydelse som ¤*, utom när den är inom "....".

```
echo "¤É"
```

beräknar inte positionsparametrarna innan de skickas till **echo**, utan kommandot får ¤1 ¤2 etc som argument, dvs det är ekvivalent med nedanstående.

```
echo "¤1" "¤2"
```

I följande tabell visas hur olika metatecknen i shell avkodas inom de olika citationsmetoderna apostrof ('....') och citationstecken (".....") samt vid kommandosubstitution (`....`).

		Metatecken					
		Ö	¤	*	`	"	'
Apostrof	' :	nej	nej	nej	nej	nej	avsl.
Citations-tecken	" :	ja	ja	nej	ja	avsl.	nej
Kommando-substitution	` :	ja	nej	nej	avsl.	nej	nej

avsl. = avslutar
ja = tolkas
nej = tolkas inte

Notera vid egna tester att kommandot **echo**, liksom flera andra kommandon, internt tolkar vissa specialtecken, t ex Ö i positionsparametrarna. Denna tolkning sker då inte av shell. Tabellen ovan visar bara om shell tolkar tecknen.

Notera även att vid kommandosubstitution den nya shell som startas kan tolka metatecken som anges. Bl a ¤ och * tolkas alltså inte av den shell som substituerar utdata till kommandosträngen. Exempelvis så kommer * i nedanstående exempel att tolkas som filerna i /usr/kalle och inte i det nuvarande aktuella biblioteket.

```
echo `(cd /usr/kalle ; ls *)`
```

I de fall där mer än en tolkning skall göras på en sträng kan det i shell ingående kommandot **eval** användas. Till exempel, om variabeln X har värdet ¤y, som i sin tur har värdet pqrrrr, medför detta att följande sekvens skriver ut värdet av pqrrrr.

```
y=pqrrrr
X='¤y'
eval echo ¤X
```

Generellt sett utvärderar kommandot **eval** sina argument precis som alla andra kommandon och resultaten behandlas som indata till shell. Indata läses och kommandot exekveras, exempel

```
wg='eval who ö fgrep'
¤wg kurt
```

är ekvivalent med

```
who ö fgrep kurt
```

I detta exempel behövs **eval** därför att det annars inte finns någon tolkning av metatecken, t ex ö, som följer på en substitution.

7.4 Fel som upptäcks av shell

Hur ett fel behandlas som upptäcks av shell beror på vilket fel det är och om shell används interaktivt eller inte. Om in- eller utdata till ett shell är kopplat till en terminal är detta shell ett interaktivt shell. Ett shell som anropas med optionen -i satt är också ett interaktivt shell även om det inte är kopplat till en terminal.

En exekvering kan misslyckas om något av följande fall inträffar:

* Omdirigeringen av in-och utdata har misslyckats, t ex en fil existerar inte eller kan inte skapas.

* Kommandot existerar inte eller kan inte exekveras.

* Kommandot avslutas på ett felaktigt sätt t ex bussfel eller minnesfel. (Se nästa sida för en lista på D-NIX systemsignaler.)

* Kommandot avslutas normalt men utgångsstatusen är skild från noll.

I alla de ovan beskrivna fallen kommer shell att gå vidare och exekvera nästa kommando. Ett felmeddelande kommer att skrivas ut av shell för alla utom det sista.

Optionen -e ser till att shell avslutas direkt om utgångsstatus från ett kommando är skilt från noll. -e har ingen effekt om shell är interaktiv.

Dessutom kan nedanstående fel inträffa,

* Syntaxfel t ex 'if...then...done', där 'done' borde vara fi.

* En signal t ex interrupt från en annan process eller från tangentbordet. Shell väntar på att kommandot skall avslutas och avslutas eller återvänder till terminalen. Kommandot **trap** används för att fånga signaler och göra något annat.

* Något fel på ett kommando som ingår i shell, t ex **cd**.

I dessa tre fall kommer shell att gå ur kommando proceduren. Ett interaktivt shell återvänder till terminalen för att läsa ett nytt kommando.

D-NIX signaler:

```
 1.     Hangup
 2.     Interrupt (DEL från tangentbordet)
 3.*    Uthopp (quit, CTRL-Ö från tangentbordet)
 4.*    Ej tillåten instruktion
 5.*    Trace trap
 6.*    IOT instruktion
 7.*    EMT instruktion
 8.*    Floating point exception
 9.     Kill (kan inte fångas eller ignoreras)
10.*    Bussfel
11.*    Segmentöverträdelse
12.*    Felaktigt argument till system anrop
13.     Skriver till pipe utan att någon läser
        det.
14.     Alarm klocka
15.     Programavslutning (från kill(1))
```

De signaler som är märkta med asterisk (*) orsakar en minnes dump om de inte fångas av kommandot **trap**. Emellertid ignorerar shell signalen UTHOPP (quit) som är den enda externa signal (från tangentbordet) som kan ge upphov till minnes dump. De signaler som är intressanta för shell är 1,2,3,14 och 15.

En process kan fånga alla signaler utom signal 9, för att hindra att processen avbryts. Detta görs med kommandot **trap**.

Signaler kan sändas till bakgrundsprogram med kommandot **kill**. Till förgrundsprocesser kan signalerna Interrupt(2) och Quit(3) sändas genom kontrolltangenter från tangentbordet. Koden för tangenterna (normalt DEL och CTRL-Ö) kan emellertid ändras, vilket också görs av många program.

7.4.1 Hanteringen av fel

En shellprocedur avslutas normalt om ett interrupt genereras från terminalen. Men om det skulle behövas kan kommandot trap användas för att ge möjlighet att 'städa upp', som t ex att ta bort temporära filer. Exempel:

```
trap "rm /tmp/ps¤¤ ; exit" 2
```

I exemplet sätts en trap för signal 2 (interrupt från terminalen). Om denna signal mottages, exekveras kommandot

```
rm /tmp/ps¤¤ ; exit
```

Notera att '....' inte kan användas eftersom ¤¤ skall avkodas av shell till processnumret.

exit är ett annat kommando som ingår i shell och som kan avsluta exekveringen av en shellprocedur. Om inte **exit** kommandot anges i trap-instruktionen skulle **trap** kommandot, efter det att trap utförts, återvända till den position där shellprogrammet befann sig innan det blev avbrutet.

D-NIX signalerna kan behandlas på ett av följande tre sätt. De kan ignoreras, de skickas då aldrig till processen. De kan bli fångade, i detta fall måste processen bestämma vilken handling som skall utföras när signalen mottages. Till sist kan tas om hand av systemet, varvid de ger upphov till en avslutning av processen utan att någon ytterligare handling måste utföras. Trap kommandot ignoreras om en signal ignoreras redan vid inträdet t ex genom att ett kommando exekveras i bakgrunden.

Hur **trap** används visas i nedanstående exempel som är en modifierad version av proceduren mytouch från kapitel 7.2.7. Filen junk¤¤ skall rensas bort om kommandot blir avbrutet.

```
flag=
trap "rm -f junk¤¤ ; exit" 1 2 3 15
for i
  do case ¤i in
      -c) flag=N ;;
       *) if    test -f ¤i
          then  ln ¤i junk¤¤ ; rm junk¤¤
          elif  test ¤flag
          then  > ¤i
          else  echo fil Ö'¤iÖ' existerar ej
          fi ;;
     esac
  done
```

Kommandot **trap** uppträder innan den temporära filen skapas, på grund av att det är möjligt att processen avbryts direkt efter skapandet av filen, innan fler kommandon hinner utföras.

Signalen 0 används till att indikera ett kommandon som skall exekveras vid utträdet ur nuvarande shell.

En procedur kan själv välja att ignorera signaler genom att nollsträngen specificeras som ett argument till trap kommandot.

```
trap '' 1 2 3 15
```

Detta ger till resultat att signalerna HANGUP, INTERRUPT, UTTRÄDE (QUIT) och KILL ignoreras av både proceduren och de kommandona som anropas från proceduren. D-NIX-kommandot **nohup** fungerar på detta sätt.

Trap återställs genom att följande skrivs. Signalerna 2 och 3 återfår i detta exempel sina defaultvärden.

```
trap 2 3
```

En lista på de för tillfället aktiva trap värdena och motsvarande kommandon kan fås genom **trap** utan parametrar.

```
trap
```

Ett **trap**-kommando behöver inte avslutas med **exit**. Utan **exit** fortsätter exekveringen av proceduren där den avbröts, efter att trap-kommandot utförts. Exemplet nedan gör att signalen tas om hand, men ingen åtgärd utförs utan prceduren fortsätter som om ingenting hänt. Kommandot **:** utför ingenting då signalen 2 kommer.

```
trap : 2
```

Proceduren nedan, som vi kallar scan, är ett exempel på en användning av **trap** där proceduren fortsätter efter trap kommandot. Scan tar varje bibliotek i det aktuella biblioteket, visar dess namn och exekverar därefter varje kommando som skrivs på terminalen med det angivna biblioteket som aktuellt bibliotek, tills ett end-of-file (CTRL-D) eller interrupt (DEL) mottages från terminalen. Vid end-of-file byter proceduren till nästa bibliotek. Ett interrupt ignoreras under exekveringen av kommandot men avbryter proceduren om scan står och väntar på att operatören ska ange nästa kommando.

```
ddd=`pwd`
for i in *
  do if test -d ¤ddd/¤i
        then cd ¤ddd/¤i
             while echo "¤i:"
                   trap exit 2
                   read kommando
             do trap : 2 ; eval ¤kommando ; done
     fi
  done
```

Här används även shell-kommandot **read**, vilket ingår i shell och läser en rad i taget från standard input och placerar resultatet i en variabel, i detta fall i variabeln 'kommando'. Kommandot returnerar en utgångsstatus som är skild från noll om det läser ett filslut, varvid proceduren går ur while-loopen. Om ett avbrott (signal 2) genereras under read-kommandot, utförs trap-kommandot exit, medan avbrott under exekvering av det inlästa kommandot ignoreras.

7.5 Exekvering av shell kommandon

För varje kommando som startas upp och som inte ingår i shell, startas en ny process upp med systemanropet 'fork'. Innan kommandot exekveras, skapar fork en 'barn'-process genom att kopiera 'fader'-processens öppna filer och signalstatus.

Även kommandon i filer med samma namn som interna shell-kommandon kan exekveras genom shell-kommandot . (en punkt) följt av ett mellanslag och kommandofilens namn.

I en del fall behöver inte en ny process startas, då används istället det i shell ingående kommandot **exec**. Detta kommando byter bara ut shell mot ett annat kommando. Om det används direkt från tangentbordet för att exekvera ett kommando kommer användaren att loggas ut efter att kommandot utförts, eftersom den ursprungliga shell ersattes av kommandot. En enkel version av kommandot **nohup** skulle kunna vara en shellprocedur med följande utseende.

```
trap ''  1 2 3 15
exec ¤*
```

trap förhindrar att de kommandon som skapas senare känner av de specificerade signalerna och **exec** ersätter den shell som exekverar nohup med det specificerade kommandot.

Det mesta om omdiregering av in-/utdata har beskrivits tidigare. Nedan är en sammanfattande tabell.

In- och utdata beräknas från höger till vänster precis som de förekommer i kommandot. Detta är av betydelse då en filbeskrivare (file-descriptor) dupliceras, dvs flera filbeskrivare används för åtkomst till samma fil. Omdirigering med > är samma som 1> för standard output och omdirigering med < är samma som 0< för standard input.

> word	Standard output (filbeskrivare 1) skickas till filen word, som skapas om den inte redan existerar.
>> word	Standard output (filbeskrivare 1) skickas till filen word. Om filen existerar läggs utdata till sist i filen, annars skapas filen.
< word	Standard input (filbeskrivare 0) tas från filen word.

<<word
eller
<<-word
Standard input (filbeskrivare 0) läses från följande rader tills en rad med enbart strängen word påträffas eller tills filslut. Detta kallas 'here-dokument'. word kan vara en valfri sträng. Om något tecken i word är citerat, t ex med Ö, kommer shell inte att tolka metatecken i here-dokumentet, annars kommer parameter- och kommandosubstitution att utföras. Önyrad ignoreras alltid. Tecknet Ö måste alltid används till att citera tecknen Ö, ¤, ` och även första bokstaven i word om strängen word i dokumentet inte ska vara avslutningstecken. Alla tab-tecken i strängen word och i dokumentet ignoreras om ett minustecken anges (<<-word).

<& digit
Filbeskrivare 0 (standard input) dupliceras med filbeskrivare digit, varvid standard input läses från den fil som associerats med filbeskrivare digit.

>& digit
Filbeskrivare 1 (standard output) dupliceras med filbeskrivare digit, varvid standard output sänds till den fil som associerats med filbeskrivare digit.

<&-
Standard input (filbeskrivare 0) stängs.

>&-
Standard output (filbeskrivare 1) stängs.

Omdirigeringarna som beskrivits ovan kan föregås av en siffra, vilket medför att den filbeskrivare som skapas kommer att specificeras av denna siffra istället för av defaultvärdena 0 (för 0<) och 1 för 1>). Som exempel kan "standard error output", som har filbeskrivare 2, sändas till en fil genom:

```
... 2> filnamn
```

Till standard error output sänder många kommandon eventuells felmeddelanden. Ett kommando som har utdata riktad mot en fil kan även sända felmeddelanden till samma fil genom:

```
... >filnamn  2>&1
```

Filbeskrivare 2 duplicerar här filbeskrivare 1, varvid utdata och felmeddelanden blandas, på samma sätt som normalt sker vid utskrift till terminalen. Notera att standard output (>) måste specificeras innan dupliceringen (2>&1) sker i kommandot.

Nya filer med andra filbeskrivare än 0, 1 och 2 kan öppnas och/eller skapas genom att kommandot **exec** används utan att någon process anges, dvs med enbart omdirigeringar. Dessa kommer då att gälla nuvarande shell. Till exempel kan en fil med filbeskrivare 5 öppnas, från vilken kommandot **read** kan läsa strängar till shellvariabler. På detta sätt kan **read** användas för att omväxlande läsa från olika filer.

```
exec  5< annanfil
read  0<&5 onestr twostr threestr
echo  'Från filen :' ¤onestr : ¤twostr : ¤threestr :
read  fourstr
echo  'Från tangentbordet :' ¤fourstr
```

I detta exempel läses en rad från filen annanfil och shellvariablerna onestr och twostr tilldelas värdet av de två första textsträngarna i raden medan threestr innehåller resten av raden. Därefter används **read** igen men läser denna gång från terminalen.

Miljön för ett kommando som körs i bakgrunden (startas med &) påverkas genom att defaultvärdet för standard input till ett sådant kommando är den tomma filen /dev/null. Detta förhindrar att shell och kommandot, som körs parallellt, läser samma indata. Om så inte vore fallet skulle snabbt kaos uppstå, t ex skulle kommandot

```
ed fil 5
```

tillåta att både editorn och shell läste samma indata vid samma tidpunkt. För utdata ändras inget automatiskt utan användaren måste omdirigera standard output om bakgrundsprocessen sänder data den vägen och dessa inte får hamna på terminalen.

Den andra modifieringen som utförs på ett kommando som körs i bakgrunden, är att signalerna UTHOPP (QUIT) och INTERRUPT 'stängs av' så att de ignoreras av kommandot. Detta medför att dessa signaler kan användas från terminalen till andra kommandon, utan att det påverkar kommandon som körs i bakgrunden. OBS! Trap kommandot har ingen effekt på en ignorerad signal.

7.5.1 Anrop av shell

Följande optioner tolkas av shell när det anropas. Dessutom finns flera shell-optioner som kan sättas och tas bort med kommandot **set**. Nedanstående optioner kan inte ändras med **set**.

Om det första tecknet i argument noll (kommandonamnet) är ett minustecken, kommer kommandon först att läsas från filerna /etc/profile och ¤HOME/.profile. Därefter läses de enligt nedan. Detta gäller t ex den shell som startas vid inloggning till systemet.

Om inte -c eller -s optionen väljs kommer första argumentet på kommandoraden att tolkas som en fil med en shell-procedur. Denna procedur utförs därvid med övriga argument som positionsparametrar.

-V	Printar versionsnumret för kommandot **sh**.
-c string	Om optionen -c är satt kommer kommandona att läsas från strängen string på kommandoraden istället för från en fil.
-s	Om optionen -s är satt eller om inga argument återstår, kommer kommandona att läsas från standard input. Med -s används övriga argument som positionsparametrar i shell. Utdata från shell kommer normalt att skrivas till filbeskrivare 2, dvs till standard error output.
-i	Om -i optionen är satt eller om in-/utdata från shell är kopplad till en terminal, är shell interaktiv. I detta fall ignoreras signalen TERMINATE (för att inte kill 0 skall avsluta ett interaktivt shell) och signalen INTERRUPT fångas men ignoreras normalt (interrupt kan utföras i kommandot **wait**). Signalen UTHOPP (QUIT) ignoreras alltid.
-r	Med optionen -r blir startas shell som en begränsad shell, där användaren hindras att utföra vissa operationer. Se nästa kapitel.

7.5.2 Begränsad shell - rsh

Shell kan startas upp som en begränsad shell med en av följande metoder:

- Shell startas med optionen -r.
- Shell startas med kommandot **rsh**, länkat till **sh**.
- Variabeln SHELL är definierad och innehåller bokstaven r.

En begränsad (restricted) shell tillåter allt som en vanlig shell utom följande operationer. Dessa kan inte göras:

- Byte av aktuellt bibliotek (**cd**).
- Ändring av variabeln PATH.
- Kommandon med namn som innehåller **/**.
- Omdirigering av utdata (> eller >>).

Detta begränsar användaren till kommandon enligt den fördefinierade PATH och i det bibliotek som är aktuellt då den begränsade shell startas. Användaren kan inte heller använda omdirigering för att skriva till några filer. Om den begränsade shell startas vid inloggning, exekveras däremot först /etc/profile och $HOME/.profile med fulla rättigheter och kan till exempel sätta upp PATH och flytta till rätt bibliotek innan användaren får ge egna kommandon.

Kommandon i kommandofiler som startas exekveras alltid med fulla rättigheter om inte shell-optionen -r anges.

7.6 Syntax och konventioner

7.6.1 Kommandosyntax

```
post:           indata och utdata
                namn=värde

enkelt          post
kommando:       enkla kommandon

kommandon:      enkla kommandon
                (kommandolista)
                kommandolista
                for namn do kommandolista done
                for namn in ord ... do kommandolista done
                while kommandolista do kommandolista done
                until kommandolista do kommandolista done
                case ord in case-del esac
                if kommandolista then kommandolista else-de

pipeline:       kommando
                pipeline ö kommando

andor:          pipeline
                andor && pipeline
                andor öö pipeline

kommandolista:  andor
                kommandolista ;
                kommandolista &
                kommandolista ; andor
                kommandolista & andor

indata-utdata:  > fil
                < fil
                >> fil
                <<ord

fil:            ord
                &siffra
                &-

case-del:       mönster) kommandolista ;;

mönster:        ord
                mönster ö ord
```

```
else-del        elif kommandolista then kommandolista
                else kommandolista
                tomt

tomt:

ord:            en sekvens av icke blanka tecken

namn:           en sekvens av bokstäver, siffror och under-
                strykningar som börjar med en bokstav

siffra:         0 1 2 3 4 5 6 7 8 9
```

7.6.2 Metatecken och reserverade ord

För de symboler som är svenska specialtecken anges inom parentes motsvarande internationella tecken.

a) syntax

```
ö       pipe symbol (Egentligen 0öA)
&&      'and' symbol
öö      'or' symbol (Egentligen 0ööA)
;       kommandoseparator
;;      case begränsare
&       bakgrundskommando
()      kommando gruppering
<       omdirigering av indata
<<      indata från ett here dokument
>       skapande av utdata
>>      utdata läggs till
=       tilldelning av värde
```

b) mönster

```
*       matcha alla tecken inklusive blanktecken
?       matcha ett ensamt tecken
Ä...Å   matcha alla tecken som står innanför
Ä!...Å  matcha alla tecken utom de som anges
        (Egentligen är tecknen 0Ä...ÅA)
```

c) substitution

```
¤ä...å  substituerar en shell variabel
        (Egentligen är tecknen 0Ä...ÅA)
`...`   substituerar utdata från kommando
        (Egentligen är tecknen 0`...`A)
```

d) citering

```
Ö       citerar nästa tecken (Egentligen 0öA)
        dvs tecknet behandlas inte som metatecken.
'...'   citerar de inneslutna tecknen utom '
"..."   citerar de inneslutna tecknen utom ¤ ` Ö "
```

e) reserverade ord

```
if then else elif fi
case in esac
for while until do done
()
:
```

8

9

9. Lp och lpr - skrivarhantering

9.1 Val av lp eller lpr skrivarsystem

I systemet skall bara en av dessa två skrivarsystem vara aktivt. Vid leverans är **lp**-systemet standardsystem och det enda användaren behöver göra är att koppla in skrivare och tillåta utskrifter till dessa skrivare (enable och accept).

Om istället **lpr**-systemet skall användas måste filen /etc/rc ändras så att **lpr**-systemet startas upp då systemet startas istället för **lp**-systemet. Dessutom skall då även kommandot **print** ändras. Detta kommando är en shell-procedur som ligger i filen /bin/print och anropar **lp**, varför **lp**-kommandot i denna fil skall ändras till ett **lpr**-kommando.

Dessa ändringar beskrivs i **IH** för datorsystemet.

9.2 Lp skrivarhantering

Skrivarsystemet LP är en serie kommandon, som hanterar en eller flera skrivare. Texter som skall skrivas ut läggs på kö (spooling) till olika skrivare och de verkliga utskrifterna sker i bakgrunden. LP kan även skriva till skrivare som finns på andra datorer, anslutna via ett nätverk. Systemadministratören kan definiera om och anpassa systemfilerna till olika typer av skrivare och grupper av skrivare kan hanteras genom att systemet självt väljer en ledig skrivare om utskriften sänts till gruppen istället för till en speciell skrivare. Detta ger maximalt kapacitetsutnyttjande i systemet. Vid utskriften kan konverteringar göras automatiskt för att anpassa formatet till den fysiska skrivaren. Användaren kan med kommandon utföra följande:

* Sända en skrivbegäran till kön eller ta bort den.

* Hindra eller tillåta skrivkö till olika enheter.

* Starta och stoppa utskrifter från köerna.

* Ändra skrivarkonfigurationen.

* Lista status för LP-systemet.

Detta kapitel beskriver hur den som är LP-administratör kan använda de särskilda kommandon som inte vanliga användare har tillgång till för att modifiera LP-systemet för bästa prestanda.

I detta kapitel anges referenser till olika kommandon och filer.

För kommandon och filer som inte beskrivs i **AH**, finns referenser med formen **name**(N), vilka refererar till andra handböcker. I de fall N är en siffra eller en siffra följd av en bokstav, refereras till handboken för utvecklingssystemet.

9.2.1 Översikt över funktioner i lp-systemet

Definitioner

Vissa termer måste definieras innan en kort beskrivning ges av de olika kommandona i LP-systemet. LP är uppbyggt för att kunna anpassas till många olika konfigurationer av datorsystem, Ändringar införs med kommandot **lpadmin**(1M).

LP skiljer på begreppet logisk skrivare och fysisk skrivare(device). En fysisk skrivare är en fysisk enhet (device) eller en fil och representeras av ett fullständigt filnamn (pathname) i filsystemet. En logisk skrivare (nedan ofta bara kallad 'skrivare') har ett logiskt namn och representerar en fysisk skrivare. Den logiska skrivaren kan vid olika tidpunkter associeras med olika fysiska skrivare. En klass av skrivare (class) är ett logiskt namn som anger en grupp logiska skrivare enligt en ordnad lista. Varje skrivarklass måste innehålla minst en skrivare. Varje logisk skrivare kan tillhöra en eller flera klasser eller ingen klass alls. En destination är antingen en logisk skrivare eller en klass av skrivare. En av destinationerna kan definieras som standardskrivare i systemet. Om inte användaren anger någon destination då kommandot **lp** används, kommer utdata att skrivas till denna standardskrivare. Om i **lp** anges en logisk skrivare sänds utskriften dit, men om en klass anges så väljer systemet den första lediga skrivaren inom den angivna klassen.

Då kommandot **lp** används skapar det en skrivbegäran, som består av de filer som ska skrivas ut samt optionsflaggorna från **lp**'s kommandorad. Ett interfaceprogram krävs för varje logisk skrivare. Bakgrundsprogrammet som hanterar utskrifterna heter **lpsched**(1M) och sänder skrivbegäran till olika destinationer genom att anropa olika interfaceprogram, vilka sköter den direkta fysiska utskriften och eventuell formatering och annan anpassning till skrivaren.

LP-konfigurationen i ett system består av fysiska enheter (device), destinationer och interfaceprogram.

Kommandon

Generella kommandon för alla användare:

Med **lp**-kommandot sänder användarna utskrifter från filer. Det skapar och sänder utskriftsbegäran till systemet och returnerar ett könummer (request-id). Formen på könumret (request-id) är

```
dest-seqno
```

och sänd till användaren. Det skrivs vanligen ut på terminalen. I könumret är 'dest' den destination dit utskriftsbegäran sändes och 'seqno' är ett unikt sekvensnummer i hela LP-systemet.

Med kommandot **cancel**(1) kan utskriftsbegäran som sänts tas bort ur kön. Även utskrifter som redan börjat skrivas ut stoppas. Som parameter anges antingen en (eller flera) logiska skrivare direkt, för att stoppa de utskrifter som pågår på dessa skrivare eller ett eller flera könummer (request-id), för att ta bort specifika utskriftsbegäran.

Kommandot **disable**(1) hindrar **lpsched** från att vidarebefordra utskrifter till en logisk skrivare.

Kommandot **enable**(1) tillåter **lpsched** att vidarebefordra utskrifter till en logisk skrivare.

Kommandon som enbart LP-administratören får använda:

En person eller flera, som fungerar som LP-administratörer, kan utföra nedanstående speciella funktioner. Det krävs antingen att de loggar in till systemet som 'super-user' eller med användarnamnet 'lp' för att de ska tillåtas utföra dessa funktioner. Alla filer i LP-systemet ägs av användaren 'lp' utom kommandona **lpadmin** och **lpsched**, vilka ägs av root. Följande kommandon kommer att beskrivas mer i detalj i följande kapitel:

lpadmin Ändrar konfigurationen hos LP-systemet. I flera fall måste lpsched stängas av medan förändringarna sker.

lpsched Hanterar köerna och sänder utskrifter till de olika interfaceprogrammet, vilka sköter den direkta utskriften till de fysiska skrivarna.

lpshut Stoppar **lpsched**. All printning stoppas temporärt, men andra kommandon i LP-systemet kan fortfarande användas.

accept Tillåter användaren att med **lp** sända utskrifter till angivna destinationer.

reject Hindrar **lp** från att acceptera utskrifter till angivna destinationer.

lpmove Flyttar köade utskrifter från en destination till en annan. Även hela kön till en destination kan flyttas till en annan. Detta kommando kräver att lpsched är stoppad.

9.2.2 Ändra lp-konfigurationen - "lpadmin"

Ändringar i konfigurationen hos LP-systemet gös med kommandot **lpadmin** och filerna skall aldrig ändras manuellt. **lpadmin** spärrar mot vissa ändringar om inte **lpsched** stoppas först. Vissa funktioner kan emellertid utföras med systemet igång. Detta anges då nedan.

Tillägg av nya destinationer

Följande information krävs för att **lpadmin** ska lägga till en ny logisk skrivare

1. Skrivarnamnet (**-p** skrivare) är ett valfritt namn som måste följa reglerna nedan:

   * Det får inte vara längre än 14 tecken.

   * Det måste bestå av enbart alfanumeriska tecken samt understrykningstecken.

   * Det får inte vara samma som en redan definierad destination (skrivare eller klass).

2. Den fysiska skrivaren (device) som associeras med den logiska skrivaren anges med (**-v** device). Detta är ett fullständigt filnamn (pathname) till en fysisk enhet eller en fil till vilken användare 'lp' har skriv-tillstånd. Den fysiska enheten kan vara en skrivarut-gång eller en login-terminal. I vissa fall används denna ej (t ex vid anslutning via ett nätverk), och kan då definieras som /dev/null.

3. Skrivarens interfaceprogram. Detta kan specificeras på ett av tre sätt.

   * Det kan väljas från en lista av modellinterface som levereras med LP-systemet. Detta görs med (**-m** model).

   * Det kan vara samma interfaceprogram som en annan redan definierad skrivare (**-e** annanskrivare).

   * Det kan vara ett eget filter-program, som skall vara i en fil som ägs av användare lp. Det väljs då med (**-i** interface).

Valfri information enligt nedan kan anges när en ny logisk skrivare skapas:

1. Optionen **-h** kan specificeras för att visa att den fysiska enheten är en skrivarutgång eller en fil. Detta antas annars som standard. Om enheten däremot är en login-terminalport, måste **-l** specificeras till **lpadmin** (bokstaven lilla l). Detta tvingar **lpsched** att automatiskt koppla bort inloggningsmöjligheten på den använda fysiska enheten varje gång **lpsched** startar. Detta rapporteras då status läses med kommandot **lpstat:**

   ```
   lpstat -pa
   printer a (login terminal) disabled Oct 31 11:15
   - disabled by scheduler: login terminal
   ```

   Detta sker för att inte någon ska logga in på den kanal där utskriften sker.

2 Den nya logiska skrivaren kan läggas in som medlem i en eller flera skrivarklasser. Detta sker med (**-c**class). Om en klass anges som inte finns, skapas den. Namnet på klassen måste vara enligt samma regler som namnet på en logisk skrivare enligt ovan.

EXEMPEL

Följande exempel kommer att ligga till grund för flera andra exempel i senare kapitel.

1. Skapa en logisk skrivare med namnet pr1, som associeras med enheten /dev/printer och välj modellprogrammet 'dumb' som interfaceprogram.

   ```
   /usr/lib/lpadmin -ppr1 -v/dev/printer -mdumb
   ```

2. Skapa en logisk skrivare med namnet pr2, som associeras med enheten /dev/tty22 och välj ett interfaceprogram som är en ändrad version av modellprogrammet prx. Döp interfaceprogrammet till pnew. /dev/tty22 är normalt en inloggningsterminal i systemet. Kopiera först modellprogrammet och ändra innan lpadmin används.

   ```
   cp /usr/spool/lp/model/prx  pnew
   ed  pnew            (Gör ändringarna!)

   /usr/lib/lpadmin -ppr2 -v/dev/tty22 -ipnew -l
   ```

3. Skapa en logisk skrivare med namnet pr3, som associeras med enheten /dev/tty23. Skrivaren pr3 skall läggas in i en ny klass som ska heta cll och använder samma interfaceprogram som skrivare pr2.

```
/usr/lib/lpadmin -ppr3 -v/dev/tty23 -epnew -ccll
```

Ändra en befintlig destination

Vid ändring av en befintlig destination används namnet för den logiska skrivaren (-pprinter) som referens. Följande kan ändras:

1. Den fysiska enheten (device) för skrivaren kan ändras med (-vdevice). Om bara detta skall ändras kan det göras med **lpsched** igång.

2. Ett annan interfaceprogram för den logiska skrivaren kan väljas (-mmodel), (-eannanskrivare), (-iinterface).

3. Skrivaren kan specificeras som direktkopplad (-h) eller som logisk terminalport (-l).

4. Skrivaren kan läggas in i en ny eller redan befintlig klass (-cclass).

5. Skrivaren kan tas bort från en klass (-rclass). Då den sista medlemmen i en klass tas bort, försvinner även klassen. Ingen destination, som det finns en kö av utskrifter till, kan tas bort. I detta fall måste **lpmove** eller **cancel** först användas för att flytta eller ta bort kön.

EXEMPEL

Dessa exempel är baserade på den LP-konfiguration som skapades i exemplet i föregående kapitel.

1. Lägg in skrivare pr2 till klassen cll.

```
/usr/lib/lpadmin -ppr2 -ccll
```

2. Byt skrivare pr2's interfaceprogram till modellprogrammet prx, ändra dess fysiska enhet till /dev/tty24 och lägg in den i den nya klassen, som heter cl2. Tillsammans med ovanstående kommer den då att tillhöra två klasser.

```
/usr/lib/lpadmin -ppr2 -mprx -v/dev/tty24 -ccl2
```

Notera att skrivare pr2 och pr3 nu använder olika interfaceprogram, trots att pr3 ursprungligen skapades med samma interfaceprogram som pr2.

3. Specificera att skrivare pr2 nu är en direktkopplad skrivare.

```
/usr/lib/lpadmin -ppr2 -h
```

4. Lägg in även skrivare prl i klassen cl2.

```
/usr/lib/lpadmin -pprl -ccl2
```

Medlemmarna i klassen cl2 är nu pr2 och prl, i denna ordning! Utskrifter som sänds till klassen cl2 kommer att sändas till pr2 om den är ledig; endast om pr2 är upptagen sänds den till nästa, dvs prl.

5. Ta bort skrivare pr2 och pr3 från klassen cll.

```
/usr/lib/lpadmin -ppr2 -rcll
/usr/lib/lpadmin -ppr3 -rcll
```

Eftersom pr3 var den sista medlemmen i klass cll, tas även klassen bort.

6. Lägg in pr3 i en ny klass som ska heta cl3.

```
/usr/lib/padmin -ppr3 -ccl3
```

Specificera standarddestination i systemet

Standard-destinationen i systemet kan ändras även medan **lpsched** är igång.

EXEMPEL

1. Ange att klassen cll skall vara standarddestination, dvs om kommandot **lp** ges utan att destinationen anges sänds utskriften till någon skrivare i klassen cll.

```
/usr/lib/lpadmin -dcll
```

2. Ta bort standarddestinationen.

```
/usr/lib/lpadmin -d
```

Ta bort destinationer i systemet

Klasser och logiska skrivare kan tas bort endast om det inte finns några köade utskrifter till dem. Köade utskrifter måste antingen stoppas helt med **cancel** eller flyttas till andra destinationer med **lpmove** innan destinationen kan tas bort. Om den borttagna destinationen är definierad som standard-destination så kommer ingen standard-destination att vara definierad efter borttagandet. Om en klass tas bort kvarstår dess tidigare medlemmar fortfarande som logiska skrivare. En klass tas även bort automatiskt om den sista medlemmet tas bort.

EXEMPEL

1. Gör skrivare prl till systemets standard-destination.

   /usr/lib/lpadmin -dprl

   Ta bort skrivare prl:

   /usr/lib/lpadmin -xprl

   Nu finns ingen standard-destination.

2. Ta bort den logiska skrivaren pr2.

   /usr/lib/lpadmin -xpr2

   Även klassen cl2 togs därmed bort eftersom pr2 var dess enda medlem.

3. Ta bort klassen cl3.

   /usr/lib/lpadmin -xcl3

   Klassen cl3 tas bort, men skrivaren pr3 finns fortfarande kvar som destination.

9.2.3 Gör en utskriftsbegäran med "lp"

Då LP-destinationer har skapats i systemet, kan användarna sända utskrifter med kommandot **lp**. Det könummer (request-id) som returneras kan användas för ett se om utskriften är klar eller för att stoppa utskriften.

LP-systemet bestämmer till vilken destination som utskriften skall ske med följande regler i ordning.

* Om användaren direkt anger destinationen -ddest på lp-kommandoraden används denna (dest).

* Om shellvariabeln LPDEST är definierad sänds utskriften annars till den destination som den anger.

* Om en standard-destination är definierad i LP-systemet sänds utskriften annars dit.

* Om ingen av ovanstående finns, blir det ingen utskrift.

EXEMPEL

1. Det finns minst fyra sätt att med kommandot **lp** sända en fil (t ex filen /etc/passwd) för utskrift till standarddestinationen i systemet.

   ```
   lp /etc/passwd
   lp </etc/passwd
   cat /etc/passwd ö lp
   lp -c/etc/passwd
   ```

   De sista tre sätten gör att en kopia av filen skrivs ut, medan det första sättet skriver direkt från den angivna filen. Om därvid filen ändras under tiden mellan lp-kommandot och då den verkligen skrivs ut, kommer ändringarna att påverka utskriften.

2. Skriv ut två kopior av filen abc till den logiska skrivaren xyz och lägg in en titel "my file" på den extra sida med användarnamn och datum som skrivs före filen. Kommandot **pr** formaterar filen till sidor.

   ```
   pr abc ö lp -dxyz -n2 -t"my file"
   ```

3. Skriv ut filen xxx på en skrivare av typen Facit 4512 via den logiska skrivaren prp. Ange optionen "-17" till interfaceprogrammet för prp för att välja antal tecken per rad. Ge ett meddelande till användarens terminal när utskriften är klar.

```
lp -w -dprp -o17  xxx
```

I ovanstående exempel är "-17" en option som ges till interfaceprogrammet för Facit 4512 genom att **lp** skalar bort tecken "o" i "-o17". Se beskrivningen av kommandot **lpadmin**.

9.2.4 Lista lp-status - lpstat

Kommandot **lpstat** listar statusinformation om utskriftsbegäran i köerna i LP-systemet, destinationer och status för utskriftshanteraren **lpsched**.

EXEMPEL

1. Lista status för alla köade utskrifter som denna användare sänt.

   ```
   lpstat
   ```

   Statusinformationen för varje utskriftsbegäran innehåller könummer (request-id), användarens loginnamn, totala antalet tecken att skriva samt datum och tid då utskriften begärdes.

2. Lista status för de logiska skrivarna p1 och p2.

   ```
   lpstat -pp1,p2
   ```

Om angiven utskriftsbegäran har sänts till en annan dator i ett nätverk, startas automatiskt shell-proceduren /usr/spool/lp/remote/lpstat, som begär information genom att exekvera lpstat på den dator till vilken utskriftsbegäran sänts.

9.2.5 Ta bort utskriftsbegäran - cancel

Utskriftsbegäran kan tas bort med kommandot **cancel**. Två olika typer av argument kan ges - könummer (request-id) eller logiska skrivarnamn. Om könummer ges tas angivna utskrifsbegäran bort ur kön och om utskrift redan pågår stoppas den direkt. Om ett skrivarnamn anges stoppas den utskrift som just pågår på den skrivaren. Båda typerna av argument kan anges samtidigt.

EXEMPEL

Stoppa den utskrift som pågår vid den logiska skrivaren xyx.

```
cancel xyz
```

Om den användare som stoppade en utskrift inte är samma som sände den till utskriftskön, sänds automatiskt ett meddelande via **mail** till ägaren av utskriftsbegäran. LP-systemet tillåter vem som helst att stoppa utskrifter för att undvika de dröjsmål som annars kan uppstå för att uppsöka en systemansvarig vid felaktigheter i utskriften.

Om angiven utskriftsbegäran har sänts till en annan dator i ett nätverk, startas automatiskt shell-proceduren /usr/spool/lp/remote/cancel, som exekverar cacncel på på den dator till vilken utskriftsbegäran sänts.

9.2.6 Tillåt/hindra sändning av utskrifter - accept och reject

lp tillåter inte utskrift till en nyskapad destination förrän LP administratören kontrollerat att den är klar att ta emot utskrifter, t ex att interfaceprogrammet är rätt, att den fysiska enheten kopplats in etc. Med kommandot **accept** anges att det är tillåtet för **lp** att sända utskrifter till destinationen.

För att temporärt hindra **lp** från att sända utskrifter till en destination kan kommandot **reject** användas. Om t ex en skrivare har flyttats eller är trasig eller om det för tillfället finns för många utskrifter i kö till skrivare, kan detta vara lämplig. Notera att utskrifter som redan är i kö till den angivna skrivaren fortsätter att skrivas ut även efter **reject**. Kommandot **accept** används sedan för att på nytt tillåta utskriftsbegäran till skrivaren.

För att se om en destination är tillgänglig för **lp** kan kommandot **lpstat** ges med optionen **-a**.

EXEMPEL

1. Gör så att **lp** vägrar ta emot utskrifter till destinationen xyz:

   ```
   /usr/lib/reject -r"printer xyz needs repair" xyz
   ```

   Alla användare som försöker skriva ut med **lp** till xyz kommer att erhålla följande felmeddelande.

   ```
   lp can not accept requests for destination "xyz"
        -- printer xyz needs repair
   ```

2. Tillåt **lp** att sända utskrifter till destinationen xyz.

   ```
   /usr/lib/accept xyz
   ```

9.2.7 Tillåt/hindra utskrifter - enable och disable

Kommandot **enable** gör att LP-systemet tillåter utskrifter till en specificerad logisk skrivare. Utskrifter tillåts enbart via interfaceprogram för tillåtna skrivare. Observera att det är möjligt att tillåta utskrifter till en skrivare (**enable**) samtidigt som man kan hindra att fler utskrifter köas till skrivaren (**reject**) med **lp**.

Kommandot **disable** är motsatsen till **enable** och hindrar all utskrift till skrivaren, oberoende av om **lp** tillåts sända utskrifter till skrivarens kö (**accept**) eller ej (**reject**). Skrivare kan behöva kopplas bort från LP-systemet av olika skäl, som fysiska fel på skrivaren, att papperet fastnat eller om man inte vill att skrivaren ska gå då ingen personal finns på kvällen. Om utskrifter pågår vid **disable**-kommandot kommer pågående utskrift att skrivas ut igen i sin helhet (från början), antingen direkt på en annan skrivare om utskriften var riktad till en klass eller på samma skrivare när den senare tillåts (**enable**). Optionen **-c** till **disable** gör att eventuellt pågående utskrift stoppas helt (jämför kommandot **cancel**). Detta är lämpligt om skrivaren ska stängas av på grund av felaktiga utskrifter.

EXEMPEL

Hindra utskrift till skrivare xyz, t ex på grund av att papperet fastnat. Andra raden nedan skrivs ut av kommandot **disable**.

```
disable -r"paper jam" xyz
printer "xyz" now disabled
```

Lista status för skrivare xyz.

```
lpstat -pxyz
printer "xyz" disabled since Jan 5 10:15 -
      paper jam
```

Tillåt skrivare xyz igen.

```
enable xyz
```

9.2.8 Flytta utskriftbegäran mellan destinationer - lpmove

Ibland behöver köade utskriftsbegäran flyttas mellan destinationer. T ex då en skrivare kopplas bort från systemet kan det vara lämpligt att flytta över hela kön av utskrifter till en annan skrivare som fungerar. Detta görs med kommandot **lpmove**. Med **lpmove** kan även specifika utskriftsbegäran flyttas till andra destinationer. **lpmove** kan enbart användas då **lpsched** är stoppad och kan alltså inte flytta utskrifter som redan håller på att skrivas ut.

Om **lpmove** har flyttat bort en skrivarkö, hindras dessutom automatiskt **lp** från att sända fler utskrifter dit (jämför kommandot **reject**).

EXEMPEL

1. Flytta hela kön med utskriftsbegäran till skrivare abc över till skrivaren xyz.

   /usr/lib/lpmove abc xyz

   Alla flyttade utskrifsbegäran kommer att byta namn från abc-nnn till xyz-nnn. Som en sidoeffekt kommer destinationen abc att inte längre tillåtas ta emot utskrifter. Ge kommandot **/usr/lib/accept** för att kunna sända till skrivaren igen.

2. Flytta utskriftern prp-222 och abc-1200 till skrivaren xyz.

   /usr/lib/lpmove prp-222 abc-1200 xyz

   De två utskriftbegäran byter därmed namn till xyz-543 och zyx1200. Numret i andra delen av namnet ändras om samma nummer redan finns i kön tidigare.

9.2.9 Stoppa och starta köhanteraren - lpshut och lpsched

lpsched (LP scheduler) är det program som hanterar de utskrifsbegäran som sänds av lp till köerna och startar upp de begärda interfaceprogrammet för den fysiska utskriften till skrivare eller andra enheter. Varje gång köhanteraren (**lpsched**) sänder en utskrift till ett interfaceprogram skrivs även ett meddelande i en log-fil, /usr/spool/lp/log. Detta meddelande innehåller användarens login-namn, det logiska skrivarnamnet och datum och tid då utskriften startades. Om en utskrift startats på nytt (exempelvis efter **disable/enable**, kan mer än ett meddelande för samma utskrift lagras. Eventuella felmeddelanden lagras också i log-filen. Varje gång processen **lpsched** startas döps log-filen om till /usr/spool/lp/oldlog och en ny log-fil skapas.

Inga utskrifter sker förrän **lpsched** har startats.

Kommandon nedan kan användas för att se om **lpsched** är igång eller ej. Den kallas LP scheduler i utskriften från **lpstat**.

```
lpstat -r
```

lpsched startas normalt via systemfilen /etc/rc då datorsystemet startas upp och är alltid aktivt tills hela systemet stängs av eller går ner i D-NIX enanvändarnivå. När **lpsched** skall startas vägrar den om filen /usr/spool/lp/SCHEDLOCK finns, vilket hindrar att flera köhanterare startas. Annars skapas denna fil då **lpsched** startas.

Ibland kan det vara nödvändigt att stoppa **lpsched** om LP-systemet skall ändras. Detta görs med kommandot

```
/usr/lib/lpshut
```

vilket direkt stoppar **lpsched**, varvid alla utskrifter stoppas. Alla utskrifter som finns i köerna stannar kvar och väntar, medan de utskrifter som redan pågår kommer att åter skrivas ut i sin helhet när **lpsched** åter startas.

För att återstarta ges kommandot

```
/usr/lib/lpsched
```

Kort efter detta kommando bör **lpstat**-kommandot ges för att se att **lpsched** verkligen kommit igång. Om inte kan det bero på att **lpsched** tidigare avslutats felaktigt utan att filen /usr/spool/lp/SCHEDLOCK tagits bort. Försök då igen med

```
rm -f/usr/spool/lp/SCHEDLOCK
/usr/lib/lpsched
```

Varefter **lpsched** bör komma igång korrekt.

All administration av LP-systemet måste ske via **lpadmin** kommandot. Att editera filer direkt medför att spool-deamonen ej startar. Detta händer t ex då man skapar en fil i "member" utan att göra detta via **lpadmin**.

9.2.10 Interfaceprogram för skrivare

Varje logisk skrivare i LP-systemet måste ha ett interfaceprogram som utför den utskriften till den fysiska enheten. Ett interfaceprogram kan vara en shell-procedur eller ett annat exekverbart program, t ex ett C-program. I bilbioteket /usr/spool/lp/model finns några modellprogram, som är shell-procedurer. Då **lpsched** startar upp en utskrift till en logisk skrivare PPP görs detta genom att interfaceprogrammet PPP startas med följande argument. Interfaceprogrammet måste ligga i biblioteket /usr/spool/lp/interface och startas med följande argument.

```
interface/PPP id user title copies options file ...
```

där

id	är könumret (request-id), vilket returnerats av **lp** då utskriftsbegäran gjordes.
user	är inloggningsnamnet för den användare som gjorde utskriftsbegäran.
title	är en valfri titelsträng som angivits som argument till **lp**-kommandot.
copies	är antalet begärda kopior, vilket angivits som argument till **lp**.
options	är en lista, separerad av blanktecken med optioner för interfaceprogrammet. Dessa specificerades med -ooptions till **lp**.
file	är fullständiga filnamnet för filen som ska skrivas ut.
....	Flera filer kan specificeras i samma begäran från **lp**-kommandot.

EXEMPEL

Följande exempel är utskrifter som gjorts av användare "smith". Standarddestination har definierats som skrivare "xyz". Varje exempel visar både **lp**-kommandot och motsvarande startkommando för interfaceprogrammet. Alla parametrar finns med, även de som är tomma strängar. I listan nedan är långa rader uppdelade på två för att få plats på sidan.

```
1.  lp /etc/passwd /etc/group
    interface/xyz xyz-52 smith "" 1 "" /etc/passwd
                  /etc/group

2.  pr /etc/passwd ö lp -t"users" -n5
    interface/xyz xyz-53 smith users 5  ""
                  /usr/spool/lp/request/xyz/d0-53

3.  lp /etc/passwd -oa -ob
    interface/xyz xyz-54 smith "" 1 "a b" /etc/passwd
```

När interfaceprogrammet startas kommer dess standard input från den tomma enheten /dev/null och både standard output och standard error output är riktat till den fysiska enhet som specificerats för skrivaren. Enheten öppnas för både läsning och skrivning om enhetens åtkomsttillstånd så tillåter. I de fall när enheten är en vanlig fil, lägges utskriften alltid till i slutet av filen.

Med dessa förutsättningar kan interfaceprogrammet formatera data på valfritt sätt och kan t ex sätta upp de rätta terminalparametrarna (jämför kommandot **stty**) överföringshastighet, antal databitar, paritet m m. Då en shell-procedur används kan följande enbart göras då enheten är öppen även för läsning:

```
stty mode ... <&1
```

Kommandot **stty** kräver nämligen att enheten anges som standard input.

När utskriften är färdig skall interfaceprogrammet avslutas med en utgångsstatus som indikerar om allt gått bra. De statuskoder som används av **lpsched** är följande:

KOD	BETYDELSE i LPSCHED
0	Utskriften gick bra.
1 till 127	Fel uppstod vid utskriften, t ex alltför många icke-skrivbara tecken). De fel som returneras på detta sätt påverkar inte andra utskrifter i kön. **lpsched** sänder ett meddelande till användaren (mail) att det blev fel vid utskriften.
större än 127	Dessa koder är reserverade för internt bruk i **lpsched** och får ej returneras av interfaceprogram.

Om fel som påverkar även andra utskrifter uppstår (t ex om ett begärt filterprogram inte finns) bör interfaceprogrammet koppla bort skrivaren med **disable**-kommandot så att utskrifterna i kön väntar. När en skrivare kopplats bort med **disable** kommer interfaceprogrammet automatiskt att avslutas genom signal 15 (jämför kommandot **kill**).

9.2.11 Att definiera fysiska enheter som lp-skrivare

Direktkopplade enheter

Som ett exempel på hur direktkopplade enheter definieras som skrivare i LP-systemet antar vi att /dev/tty15 ska definieras som skrivare xyz. Följande urföres av den systemansvarige, inloggad som super-user:

1. Se till att LP-processen kan skriva till enheten och hindra andra processer att skriva dit.

   ```
   chown  lp  /dev/tty15
   chmod  600 /dev/tty15
   ```

2. Ändra /etc/inittab så att tty15 inte längre aktiveras som inloggningsterminal. Detta gör att processen /etc/getty inte tillåter någon att logga in på denna terminalingång. Ändra tredje fältet i raden för tty15 till strängen 'off' enligt nedan.

   ```
   t15:2:off:nice -16 /etc/getty tty15 1200
   ```

   Ge sedan kommandot:

   ```
   telinit Q
   ```

   Om en /etc/getty-process redan är igång för inloggning på tty15 tas den bort med kommandot **kill -9** med process-id som argument. Detta kan ses med kommandot **ps -xl**. När D-NIX-systemet startas upp på nytt kommer tty15 att initieras med standardparametrar, varför det är skrivarens interfaceprogram i LP-systemet som måste sätta upp rätt överföringshastighet (baud-rate) och andra moder enligt **stty**-kommandot.

3. Skapa den logiska skrivaren 'xyz' i LP-systemet och använd modellprogrammet prx som interfaceprogram.

   ```
   /usr/lib/lpadmin -pxyz -v/dev/tty15 -mprx
   ```

4. När xyz skapas kommer den inte att vara tillgänglig (disable) och **lp** kan inte sända utskrifter till den (reject). För att tillåta att utskrifter köas med **lp** till xyz, ges kommandot:

   ```
   /usr/lib/accept  xyz
   ```

Därefter tillåter LP-systemet att en kö med utskrifter får byggas upp till denna skrivare, för att skrivar ut när skrivaren gjorts tillgänglig med **enable**-kommandot:

5. Innan kommandot **enable** ges för xyz kontrolleras att utskrifter verkligen kan göras till den fysiska skrivaren. Detta innebör t ex att papperets överkant är rätt justerat (top-of-form) och att skrivaren är påslagen och är on-line. Utskrifter till skrivaren möjliggörs med:

   enable xyz

   Nu kan utskrifter som sänts till den logiska skrivaren skrivas ut.

Inloggningsterminaler

Utskrift med LP-systemet kan även ske till en inloggningsterminal. Ska t ex en Facit 4512 skrivare kopplas till en terminalport och utskrift ska ske till denna med det logiska skrivarnamnet abc, görs följande:

1. Skapa den logiska skrivaren abc med modellinterfaceprogrammet dumb:

   /usr/lib/lpadmin -**p**abc -**v**/dev/null -**m**dumb -l

   Vi anger här /dev/null som fysisk enhet eftersom den verkliga enheten kommer att specificeras varje gång abc görs tillgänglig med **enable**. Den verkliga enheten kan vara olika vid olika tillfällen. När abc skapas kommer den inte att vara tillgänglig och **lp** kommer inte heller att kunna sända utskrifter till den. För att tillåta att utskrifter köas till abc ges kommandot:

   /usr/lib/accept abc

   Därefter tillåts att utskrifter köas till abc för att senare skrivas ut när skrivaren gjorts tillgänglig. Detta bör emellertid inte ske förrän följande har utförts.

2. Utför inloggning på den önskade terminalingången, om detta inte redan gjorts.

3. Använd kommandot **tty** för att se vilken fysisk enhet terminalen är kopplad till, exempelvis /dev/tty02. Associera denna enhet med den logiska skrivaren abc med kommandot:

   /usr/lib/lpadmin -pabc -v/dev/tty02

   Obervera att kommandot **lpadmin** enbart kan användas av LP-administratören (användare lp) eller super-user. Om vanliga användare ska få skriva ut på detta sätt bör LP-administratören skapa ett program, ägt av lp eller root, som utför detta med tillståndsbiten 'set-user-id' satt.

4. Efter att skrivaren kopplats till terminalporten, eller till terminalens skrivarport kan utskrifter tillåtas med **enable**. Innan dess bör kontrolleras att utskrifter verkligen kan göras till den fysiska skrivaren. Detta innebör t ex att papperets överkant är rätt justerat (top-of-form) och att skrivaren är påslagen och är on-line. Utskrifter till skrivaren möjliggörs med:

   enable abc

   Nu kan utskrifter som sänts till den logiska skrivaren skrivas ut.

5. När alla utskrifter är klara till abc eller när terminalporten åter ska användas som login-terminal, hindras utskrifterna med:

   disable abc
   printer "abc" now disabled

   Om skrivaren abc är tillgänglig (enabled) när D-NIX-systemet startas upp eller då **lpsched** startas, kommer den automatiskt att kopplas bort (disable).

9.2.12 Sammanfattning

De administrativa rutinerna som LP-administratören behöver göra har beskrivits i detalj. Dessa funktioner är skapa och ändra skrivarkonfigurationer för LP, skapa/ändra/använda interfaceprogram för skrivare, tillåta/hindra köandet av utskrifter, tillåta/hindra utskrifter till skrivarna, flytta eller ta bort köade utskriftsbegäran, stoppa/starta köhanteraran **lpsched**. LP-systemet ger många fördelar gentemot vanliga program för skrivarhantering, bl a:

* Skrivare kan grupperas i klasser.

* LP-systemet kan konfigureras om för att anpassas till den omgivning den ska arbeta i.

* Administratören kan förse systemet med olika interfaceprogram för att formatera utskrifterna på valfritt sätt.

* Funktionerna i LP-systemet utförs med enkla kommandot och inte genom att editera filer manuellt.

9.3 Lpr skrivarhantering

Ett alternativt skrivar-system, LPR-systemet, finns med i grundpaketet. Kommandot **lpr** kan användas för att sända utskrifter till "logiska skrivare" i systemet (jämför **lp**-kommandot). Det är då bakgrundsprocessen **/usr/lib/lpd** som sköter utskriften från den kö som bildas med **lpr**. Notera att endast en av dessa båda system (**lp** eller **lpr**) skall vara aktiva i systemet.

* I LPR-systemet väljer användaren med optionen -t till lpr-kommandot ett eller flera villkor som tolkas av systemet för att utskriften skall sändas via en logisk skrivare till en lämplig fysisk skrivare. En fysisk skrivare kan ha flera logiska namn med olika styrprogram för olika ändamål.

* Initieringsprogram kan automatiskt startas upp vid varje utskrift, olika till varje logisk skrivare.

* Inledande datafiler (header) samt avslutande datafiler (tailer) kan definieras för varje logisk skrivare.

* Filter-program kan automatiskt användas för att erhålla grafik eller standardiserad kodning av styrtecken för varje logisk skrivare.

Detta kapitel beskriver hur den som är LP-administratör kan konfigurera de olika filerna i systemet för att modifiera LP-systemet för bästa prestanda.

9.3.1 Översikt över funktioner i lpr-systemet

LPR-systemet är uppbyggt för att kunna anpassas till många olika konfigurationer av datorsystem.

LPR skiljer på begreppen logisk skrivare, villkor och fysisk skrivare (device). En fysisk skrivare är en fysisk enhet (device) och representeras av ett enhetsnamn i bilbioteket /dev i filsystemet. En logisk skrivare har ett logiskt namn och en fysisk skrivare kan ha flera olika logiska namn.

Olika villkorssträngar kan associeras med varje logiska skrivare, t ex kan de skrivare som klarar grafik markeras med 'graphic' och de som klarar skönskrift markeras med 'pretty'. Dessa villkor anges som alternativ till det logiska skrivarnamnet i optionen -t till lpr-kommandot, varvid systemet väljer den skrivare som för tillfället är ledig, men som klarar av det som användaren vill kunna göra.

Utan angivande av logisk skrivare riktas utskriften till /dev/lp. Det som ska skrivas kan även riktas till **lpr**-kommandot med en pipe, varefter det läggs i skrivarkön.

LPR-systemet konfigureras genom filer i biblioteket **/usr/lib/lpdpar**.

Exempel: Skriv filen 'myfile' till skrivaren pr1:

```
lpr -tpr1 myfile
```

Filen lpdtble:
Kopplingen mellan "logisk skrivare" och fysisk skrivare görs i filen lpdtable. Denna fil innehåller en rad för varje fysisk skrivare, med innehåll som kan vara enligt följande exempel.

```
lp    myhead     pr1
lp2   head2      pr2 pretty
```

där parametrarna är (med exempel ur andra raden):

```
lp2      /dev/lp2 är den motsvarande fysiska skrivarpor-
         ten.
head2    är det logiska skrivarnamnet. Det anger bl a att
         filen /usr/lib/lpdpar/lpdhandler/head2 skrivs ut
         i början av varje dokument, om inte optionen -h0
         anges till lpr för att hindra detta. En fysisk
         skrivare kan återkomma på flera rader som därmed
         refereras med olika logiska skrivarnamn.
pr2      pr2 och pretty är villkorssträngar. Flera villkor
pretty   kan ges för varje skrivare.
```

Den andra parametern anger den logiska skrivaren. För varje logisk skrivare kan filer i följande bibliotek skapas, med samma namn som den logiska skrivaren. Om motsvarande fil saknas används inte den funktionen.

```
Biblioteket devinit:     Initieringsprogram
Biblioteket lpdheader:   Inledande datafiler (header)
Biblioteket lpdtailer:   Avslutande datafiler (tailer)
Biblioteket lpdhandler:  Filterprogram för grafik m m
```

Kommandon

Kommandot **lpr** används för att sända utskrifter till skrivarkön och användaren anger då villkor för den skrivare som skall användas. Med **lpr** kan en eller flera filer sändas till utskrift. Om inga filer anges tas data från standard input. **lpr** skapar och sänder utskriftsbegäran till systemet. Varje begäran får ett könummer och tilldelas en utskriftsprioritet. Systemet försöker effektivisera genom att skriva ut korta filer före långa.

/usr/lib/lpd är den process som sköter utskriften från kön till de fysiska skrivarna när **lpr**-kommandot används.

Kommandot **queue** listar vilka utskrifter som ligger i kö för att skrivas ut med **lpr**-spoolersystemet. Med **queue** kan icke påbörjade utskrifter tas bort ur kön. Utskriftsordningen styrs av en utskriftsprioritet som kan ändras med kommandot **queue**.

9.3.2 Ändra lpr-konfigurationen

ANM! På systemet finns två olika spoolersystem, **lp**-systemet och **lpr**-systemet. Endast en kan användas åt gången. Se installationsmanualen för systemet för detaljer.

För att återstarta utskrift av eventuella filer i utskriftskön efter systemstart, måste filen /etc/rc ha kommandon enligt nedan. Sista raden startar spool-programmet. Dessutom måste biblioteket /usr/spool/lpd vara tillgängligt för att **lpr**-systemet skall kunna användas. Detta biblitek måste vara skapat med läs/skriv- och exekveringstillstånd för samtliga användare.

```
rm -f  /usr/spool/lpd/ERRLOG
rm -f  /usr/spool/lpd/lock
rm -f  /usr/spool/lpd/lpdctl
nice -20 /usr/lib/lpd  &
```

Konfigurering skrivarval (Fil: lpdtable)

Filen /usr/lib/lpdpar/lpdtable innehåller en tabell över de skrivare som finns tillgängliga samt de inställningar som gäller för dem. Om filen inte finns tillgänglig används /dev/lp som skrivare.

Typsträngen (eller strängarna) angivna i -t optionen för **lpr** kommandot jämförs med strängarna i denna tabellfil och vanligen väljs en skrivare för vilken alla dessa strängar matchar. Därtill kan den första raden i tabellfilen specificera strängval för vilket endast en sträng behövs för att ange en grupp av skrivare. Om flera skrivare uppfyller kraven vid valet, används den skrivare som är ledig för tillfället. En default typsträng kan defineras i shellvariabeln PRTYPE. Om så inte är fallet, väljs den skrivare som är markerad med valsträngen 'default'.

Om endast /dev/lp används i systemet och ingen initiering, inga titelsidor, tailers eller filtreringsprogram används, är inte biblioteket /usr/lib/lpdpar nödvändigt.

Ett exempel på en tabellfil ges på nästa sida.

Om en rad i tabellen börjar med en sträng, tolkas denna sträng (devi ce) som den fysiska utskriftsporten och den andra strängen (tabellnamn) används för att välja initieringsprogram, titel- och tailerfiler och (-f option) filtreringsprogram. På följande strängar används som valsträngar. Tabellnamnsträngen används som valsträng för skrivaren och jämför denna med typsträngen angiven i -t optionen.

Om en rad börjar med ett mellanrum eller en TAB, tyds detta som en fortsättning på föregående rad, förutsatt att det inte är första raden på filen.

Om första raden i tabellfilen börjar med ett mellanrum eller en TAB, kommer alla strängar på denna raden att specificera valsträngar för vilket endast en strängs överensstömmande är nödvändig. Se exempel 5 ovan och exempel 3 nedan på val.

Exempel på en tabellfil med följande fält:

Device, Tabellnamn, Valsträngar.

```
          room1    room2    room3
lp      facit4542    english medium default room1
lp1     facit4544    english graphic medium default room3
lp2     diablo       swedish pretty room1
lp3     graf2        english graphic room2
```

Denna tabellen definerar /dev/lp till en Facit 4542 att vara default-skrivare med medium hastighet och ett engelskt tecken-set På samma sätt som /dev/lp1 är en Facit 4544 med ett engelskt teckenset, grafisk presentation, medium hastighet och liksom en defaultskrivare. /dev/lp2 är en diablo skrivare med ett svensk tecken-set. /dev/lp3 är en andra skrivare för grafik. I detta tabellexempel, kan gruppvalsträngarna på första raden beskriva den fysiska platsen för skrivarna.

Den andra strängen, kallad tabellnamn ovan, anger filnamn för titeloch tailer-filer, initieringsprogram för utskriftsenheter och filtreringsprogram.

Om endast /dev/lp används i systemet och ingen initiering, inga titlar, tailers eller filtreringsprogram används, behövs inte biblioteket /usr/lib/lpdpar.

Valexempel 1: Användaren specificerar optionen '-tenglish' för kommandot **lpr** och utskriften dirigeras till första fria enhet som har 'english' som angivet tecken-set. I detta fall gäller det både för /dev/lp, /dev/lp1 och /dev/lp3.

Valexempel 2: Skrivartypen kan också specifieras med en kombination av sättningar, såsom '-tenglish,graphic', vilket i detta fallet skulle betyda /dev/lp1 eller /dev/lp3. Observera att komma (,) krävs mellan varje sträng i optionen.

Valexempel 3: Om valet '-tgrapcic,rum1,rum2' är specificerat väljs vilken grafisk skrivare som helst antingen i rum1 ELLER rum2. I detta fallet kan endast /dev/lp3 användas. Detta är ett exempel där första raden i tabellfilen måste användas eftersom både 'rum1' och 'rum2' annars skulle krävas.

Valexempel 4: Typen kan också specificeras av enhetsnamnet, dvs. '-tlp', '-tlp1', '-tlp2' eller -tlp3', i vilket fall som helst kan dock inga andra valsträngar kan specificeras.

Observera att strängarna inte översätts utan endast jämförs med strängar givna som typer.

Initieringsprogram för skrivare (Bibliotek: devinit)

'Tabellnamn'-strängen i tabellfilen är namnet på en fil i biblioteket /usr/lib/lpdpar/devinit. Denna fil exekveras som kommando med en standard input och output dirigerad till skrivaren innan någon annan text skickas ut. Om denna fil inte finns, sker ingen initiering.

Titel-/tailerfiler till skrivare (Bibliotek: lpdheader och lpdtailer)

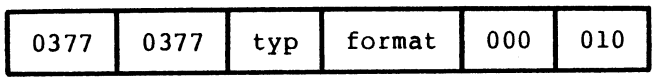
'Tabellnamn'-strängen i tabellfilen är namnet på en fil i biblioteket /usr/lib/lpdpar/lpdheader. Datan i denna filen skrivs alltid ut före annan text. Titelfilen kan innehålla setup strängar för skrivaren. Dessa behövs för att kontrollera att skrivaren alltid skriver likadant. Om ingen fil finns med detta namn, skrivs ingen titelsida ut.

'Tabellnamn'-strängen i tabellfilen är namnet på en fil i biblioteket /usr/lib/lpdpar/lpdtailer. Datan i denna filen skrivs alltid ut efter all annan text. Om ingen fil med detta namn finns, skrivs ingen särskild slutsida ut.

Filtreringsprogram för skrivare (Bibliotek: lpdhandler)

'Tabellnamn'-strängen i tabellfilen är namnet på en fil i biblioteket /usr/lib/lpdpar/lpdhandler. Denna fil innehåller filtreringsprogram, som används för avkodning av speciella sekvenser vid data output av grafisk presentation och andra speciella utskrifter. Detta är möjligt bara om -f optionen är given och om filen för filtreringsprogrammet existerar. Filtreringsprogrammen läser från standard input (= datan som skall skrivas ut) och skriver till standard output (blir automatiskt dirigerad till skrivaren) Speciella sekvenser i datainnehållet startar och stannar filtreringsprogrammet.

Den speciella sekvensen startar med två bytes med de oktala ASCIIvärdena 0377. Sekvensen ser ut som nedan och åtföljs av alla data som skall till output genom filtret.

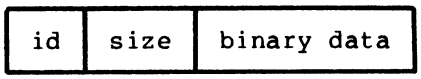


Fältet 'typ' är en byte med '0' (ASCII 060 oktalt) för textmod och '1' (ASCII 061 oktalt) för binär mod. Fältet 'format' är en sträng som sänds som kommandoradsargument till filterprogrammet. 'format' kan innehålla flera strängar separerade med mellanslag.

Om 'typ' är '0', sänds all text up till nästa speciella sekvens till filtret. För att avsluta filtreringsprogrammet kan en sekvens sändas med 'typ' som är '0', och 'format' lika med strängen 'text'.

Om 'typ' är '1' kommer en eller flera binära records att följa den speciella sekvensen. Filtreringssprogrammet avslutas när ett binärt record som har sitt 'id'-fält lika med 1 är utskrivet.

Det binära recordet har följande format:



Fältet 'id' har normalt värdet 0 (ASCII 000) om flera records följer. Det sista binära recordet skall ha 'id'-värdet 1 (ASCII 001) för att filtreringsprogrammet skall avslutas när binära data i recordet har skrivits ut. Fältet 'size' är ett 16 bitar binärt värde som har den högsta byten först. Fältet 'binary data' är en sträng med dimensionen 'size' bytes och denna strängen sänds genom filtreringsprogrammet till skrivaren.

Specificera standardskrivare i systemet

Standard-skrivare i systemet kan ändras genom att shellvariabeln PRTYPE definieras i användarens fil .profile eller i den allmänna systemfilen /etc/profile. Om denna variabel inte är definieras anses /dev/lp vara standardskrivare. PRTYPE kan innehålla en hel standardsträng med flera villkor om den innesluts mellan apostrofer. Denna används då som standard -t option för lpr-kommandot.

Villkorssträngen 'default' är reserverad och anger vilken av flera skrivare med samma villkor som är standardskrivare om varken PRTYPE eller optionen -t angivits.

9.3.3 Gör en utskriftsbegäran med "lpr"

Då LPR-filerna har skapats i systemet, kan användarna sända utskrifter med kommandot **lpr**. Varje utskriftsbegäran får ett könummer, som kan användas tillsammans med kommandot **queue** för ett se om utskriften är klar eller för att stoppa utskriften.

LP-systemet bestämmer till vilken skrivare som utskriften skall ske med följande regler i ordning.

* Om användaren direkt anger fysiska enheten -tdest på lpr-kommandoraden används denna (dest).

* Om användaren anger en logisk skrivare -tskrivare används denna.

* Om användaren anger ett eller flera villkor, varav egentligen även det logiska skrivarnamnet behandlas som ett villkor, söker systemet efter den första hittade lediga skrivaren som uppfyller alla villkor.

* Om shellvariabeln PRTYPE är definierad sänds utskriften annars till den skrivare som uppfyller de villkor som angivits i PRTYPE (motsvarande -t¤PRTYPE).

* Annars sänds utskriften till /dev/lp, som är standardskrivare om ingenting alls angivits. Detta innebär att lpr-kommandot fungerar även utan filerna i /usr/lib/lpdpar-biblioteket, men då enbart till /dev/lp och enbart om optionen -t inte används.

  * Om villkor anges i optionen -t men dessa inte kan uppfyllas och filen lpdtable finns, blir det ingen utskrift.

EXEMPEL

Exempel 1:

```
lpr -tpretty mytext
```

Sänder filen mytext till spoolkön för utskrift till en skrivare med villkorssträngen 'pretty'. Enligt exemplet ovan av filen /usr/lib/lpdpar/lpdtable, skrivs den ut på den fysiska skrivaren /dev/lp2.

Exempel 2:

```
lpr  -tenglish,graphic,rum1,rum2  myfile
```

Filen sänds till spoolkön för en skrivare som är vald av både 'english' och 'graphic', och antingen i rum1 eller rum2, förutsatt att exemplet på tabellfilen nedan är använt. För tabellfilen nedan, används den skrivare som är ansluten till /dev/lp3.

Example 3:

```
lpr  -f -tgraphic  myfile
```

Filen sänds till spoolkön för en skrivare som är vald genom strängen 'graphic'. Om det finns ett filterprogram i biblioteket /usr/lib/lpd-par/lpdhandler för den aktuella skrivaren, kommer det att användas. I varje fall ersätts TAB:ar i textsträngarna till mellanrumssträngar före output. Med tabellfilen ovan används den skrivare som är ansluten antingen till /dev/lp1 eller /dev/lp3.

9.3.4 Lista lpr-status - queue

Kommandot **queue** listar de filer som är köade i skrivarkön med kommandot **lpr**. Listan ser ut som i följande exempel:

```
DEVICE     JOBID    PRI    UID     COMMENT

/dev/lp      227     40    root    Spooled ...
             228     40    bin     Spooled ...
```

Listan är sorterad i utskriftsordning (PRI) och listan kan innehålla flera fysiska enheter (device) i den ordning de listas i filen lpdtable. Optionen -t kan användas med **queue** för att lista utskrifter till en enhet eller en skrivare motsvarande **lpr**-optionen -t.

Utskriftsordningen kan ändras genom att ändra utskriftsprioriteten, med optionen -p till **queue**-kommandot. Värdet som anges är hur mycket prioriteten skall ändras. Super-user kan även ange negativa värden, vilket innebär att prioriteten höjs istället för sänks i förhållande till övriga köade utskrifter.

EXEMPEL

Exempel 1:

Utskriftsprioriteten för utskrift 228 ska sänkas med 10.

```
queue -p 10 228
```

9.3.5 Ta bort utskriftsbegäran - queue -r

queue kan användas för att ta bort utskrifter som väntar i kön genom optionen -r. Det är däremot inte möjligt att stoppa en redan pågående utskrift och en vanlig användare, som inte är super-user, kan inte ta bort en annan användares utskrifter ur kön,

EXEMPEL

Exempel 1:

En utskrift ska tas bort ur kön. Genom ett tidigare queue-kommando har vi sett att utskriften har könummer 228.

```
queue -r 228
```

9.3.6 Stoppa och starta köhanteraren - /usr/lib/lpd

/usr/lib/lpd (LPR scheduler) är det program som hanterar de utskriftsbegäran som sänds av **lpr** till kön. /usr/lib/lpd hanterar den fysiska utskriften till skrivare eller andra enheter. /usr/lib/lpd startas normalt via systemfilen /etc/rc då datorsystemet startas upp och är alltid aktivt tills hela systemet stängs av eller går ner i enanvändarnivå. När /usr/lib/lpd skall startas vägrar den om filen /usr/spool/lpd/lock finns, vilket hindrar att flera köhanterare startas. Annars skapas denna fil då /usr/lib/lpd.

När köhanteraren startas bör den alltid startas som en bakgrundsprocess genom att tecknet & ges efter kommandot. Exempel ur filen /etc/rc:

```
/usr/lib/lpd &
```

10

11

11. Säkerhetskopiering, backup

11.1 Inledning

Av största vikt är att se till man har släpande backuper, d v s sparar band så att man har möjlighet att återskapa filer som är ett par månader gamla. I många fall upptäcks det att filer är förstörda eller raderade av misstag först efter relativt lång tid.

Om backup tas dagligen skall t. ex. fredags backuperna sparas 4 veckor. Varje månad läggs en backup undan som sparas t. ex. 6 månader. Detta kräver då:

```
måndag - torsdag    4 band
fredagar            4 band
6 månader           6 band

Totalt             14 band  => 6.500 kr
```

Denna kostnad skall vägas mot eventuell förlust av data som oftast orsakar dryga utgifter i form av stillestånd och konsultarvoden.

Följande kan inträffa:

En kund förlorar en mimer databas på 20 Mbyte. (det motsvarar cirka 3 manmånaders arbete)

Backuperna utförs som totalbackuper med ett band för måndag, ett för tisdag o s v fram till fredag. En dag uppstår ett diskfel som medför att en sektor i mimerdatabanken blir oläsbar. Databanken blir i och med detta fel oanvändbar.

Backupen som görs med tar avbryts när kommandot kommer till den den dåliga sektorn på disken. Av en slump har den systemansvarige semester denna vecka, och någon kontroll av att backupen gått utan problem, görs inte. Efter fem dar har samtliga användbara backuper av databanken skrivits över med den förstörda!

KOM IHÅG: Den dag Ni råkar ut för förlorat data kommer Ni att uppskatta ordentliga backuper.

11.1.1 Backuper på DS90 System

Backuper är mycket viktiga i en datorinstallation.

Det finns många sätt att uföra backup på ett D-NIX system. Men oberoende av metoden, så skall backuper tas för att försäkra sig om att man kan återfå all viktig information ifall systemet kraschar eller filer förloras/förstörs.

Det är betydligt vanligare att förlora data än man tror. Ofta beror det på olyckshändelser och ovarsamhet från användare. Men även system administratören (root eller su) kan lätt förlora data med katastrofala följder. Man kan även förlora data när systemet kraschar.

Ju fler användare man har i systemet desto större potentiell risk har man för krascher. Och kom ihåg det finns inget krasch-säkert system.

En lämplig backup teknik för systemet beror på flera faktorer. En snabb och enkel metod kan resultera i att man måste ägna timmar åt att återskapa systemet. Den tid man ägnar åt backup vägs emot den tid det tar att reparera systemet.

De två huvudalternativen för backup är:

Total backup:	backup av hela filsystemet även D-NIX.
Del backup:	Alla metoder där man sparar mindre än hela filsystemet.
Variationer:	Hela systemet förutom D-NIX system filer, men inkluderande filer som ändras ofta som: /etc/passwd /etc/group /etc/rc /etc/ttys Inkrementell backup. Man sparar filer baserat på det datum då filen ändrades senast.

Backup rutinerna måste anpassas till tillgängligt backup media. Om systemet har magnetbands station eller streamer tape så kan en helt annan teknik användas än om man bara har floppy disketter.

På ett system med streamer eller magnet band kan följande förslag till backup rutiner användas.

1. Efter varje större förändring i systemet görs total backup. (ex ny version av någon programvara)

2. Del backuper gjorda varje natt och återanvända vecka för vecka.

3. Total backup varje vecka (fredag). Sparas 4 veckor.

4. Total backup varje månad. Sparas minst 6 månader.

Följande shell skripts (/usr/adm/fbackup) kan användas för total backup

```
# /usr/adm/fbackup
# shell script to do full backup
# logg info is stored in /usr/adm/backuplog
# DataIndustrier DIAB AB 1986
# BLOCK is block factor used
# TSIZE is tape size in Kbytes (60000 means 60 Mbyte)
BLOCK=300
TSIZE=60000
date >/usr/adm/backuplog
cd /
tar cvfbkS /dev/st0 ¤BLOCK ¤TSIZE . Ö
    >>/usr/adm/backuplog 2>&1
date >>/usr/adm/backuplog
```

Denna skript kan startas i bakgrunden eller, om **cron** kommandot finns i systemet, från **crontab**.

Om total backupen skall startas fredag kväll vid 23:00, lägg till följande rad till crontab.

```
0 23 * * 5 /usr/adm/fbackup
```

Följande skriptar kan användas för inkrementell backup:

```
# /usr/adm/ibackup
# incremental backup skript
# DataIndustrier DIAB AB 1986
BLOCK=300
TSIZE=60000
case ¤# in
        1) days=¤1;;
        *) echo "Usage: ibackup n - where n is number of Ö
           days back"
esac
case ¤days in
        Ä1-9Å) ;;
        Ä1-9ÅÄ0-9Å) ;;
        *) echo "Bad argument, should be numeric"; exit;;
esac
echo "backing up last ¤days day(s)">/usr/adm/backuplog
cd /
find . -type f -mtime -¤days -print >/tmp/backupfiles
# if some parts of the file system should not be backuped
# change the file /tmp/backupfiles here. Example:
# cat /tmp/backupfiles ö /usr/adm/filter >/tmp/backupfilx
# mv /tmp/backupfilx /tmp/backupfiles
date >>/usr/adm/backuplog
tar cvfbkF /dev/st0 ¤BLOCK ¤TSIZE /tmp/backupfiles Ö
    >>/usr/adm/backuplog 2>&1
date >>/usr/adm/backuplog
rm -f /tmp/backupfiles
```

Shell skripten som undantar vissa filer från backup, kan se ut så här:

```
# Shell skript to remove unwanted files from backupfile
# list
# DataIndustrier DIAB AB 1986
# /usr/adm/filter

while true
do
  read f
  if test ¤f
  then
    x=`dirname ¤f`
    case ¤x in
#       here you specify directories NOT to be backuped
        ././usr/spool*) ;;
        ././tmp) ;;
        *ö.*)
              case ¤f in
#                 here you specify file names NOT to be
#                 backuped
                  ././etc/utmp) ;;
                  *ö.*) echo ¤f;;
              esac
    esac
  else
    exit;
  fi
done
```

11.2 Backup - bup

NAMN

bup - säkerhetskopiering av valda filer (backup)

SYNTAX

bup (-V) -(mwdak) filnamn ....

bup (-V) -(mwdak) -r findargument

bup (-V) (-n(mwd))

bup (-V) -l (-uvolymnamn) (filnamn ...)

bup (-V) -b(mwda) (-v) (-uvolymnamn) enhet

bup (-V) -x (-v) enhet filnamn ....

BESKRIVNING

Kommandot **bup**, som är ett hjälpprogram, utför periodiskt återkommande säkerhetskopiering av specificerade filer. **bup** använder **tar**-kommandot för att göra själva säkerhetskopieringen.Periodvis kan kommandot **cron** användas för automatisk regelbunden säkerhetskopiering. **bup** används också till att återladda filer från kopieringsmedierna.

De medier som används för säkerhetskopieringen (band och disketter) ges volymnamn. En loggfil underhålls vilken innehåller filnamn, säkerhetskopieringens volymnamn och tidpunkten för säkerhetskopieringen för alla filer på på kopieringsmediet. Loggfilen heter /usr/lib/bup/log. Då **bup** används för att återladda valda filer kommer kommandot att informera användaren om från vilket band eller diskett som filen skall tas. Volymnamn som är standard kan specificeras i filen /usr/lib/bup/volumename och de kommer att växla automatiskt. Loggfilen eller valda delar av den kan tas fram och visas med kommandot **bup-l**.

Kommandot skall utföras i två steg. Först körs **bup** för att lagra namnen på alla filer (eller bibliotek) för säkerhetskopiering på filen /usr/spool/bup/cmd. Sedan görs själva kopieringen med **bup**-kommandot och ett av följande tillval; -bd,-bw,-bm eller -ba, valfritt regelbundet aktiverat genom att innefatta kommandot i filen /usr/lib/crontab i **cron**.

För att specificera fil- resp. biblioteksnamnen körs **bup** med ett av följande tillval; '-m', '-w', '-d' eller '-a'. -m används för filer som kopieras månatligen, -w för de som kopieras veckovis, och -d för sådana som kopieras dagligen. -a används bara då filer skall slutlagras på ett externt medium och sedan tas bort ur systemet. **bup** kommer att lägga till fil- eller biblioteksnamnet i bup:s kommandofil /usr/spool/bup/cmd för senare användning då själva säkerhetskopieringen äger rum.

Fil- eller biblioteksnamnet kan specificeras antingen som normala sökvägsnamn, wildcards eller argument till ett **find**-kommando (bup -r option) för att välja filer. (Wildcard är ett specialtecken som kan stå för ett eller flera tecken i filnamnet. D-NIX använder '?' och '*' som wildcards.) **bup** kommer alltid att lägga det gällande biblioteket till vilket sökvägsnamn som helst som ges på kommandoraden och som inte är ett fullt sökvägsnamn. Om namnet på ett bibliotek anges tages hela biblioteket. Om wildcards eller **find**-argument ges kommer de inte att bli värderade förrän det är dags för själva säkerhetskopieringen förutsatt att de är omgivna av apostroftecken, (t ex '*.doc'), vilka skall hindra shell att avkoda dessa specialtecken.

Filer kan tas bort från kommandofilen med hjälp av kommandot **bup -k**. kommandofilen visas med det normala **cat**-kommandot.

```
cat /usr/lib/bup/cmd
```

Filen /usr/lib/bup/volymnamn skall innehålla tre rader med namn på den dagliga, veckovisa och månatliga säkerhetskopieringen. Raderna är märkta d:, w: eller m: och volymnamnen är skilda åt med ett kommatecken och valfritt antal mellanslag. Volymnamnen kommer automatiskt att växlas under användningen, dvs första namnet på raden kommer att användas vid nästa kopiering och detta namn kommer att flyttas till slutet av raden. Inget mellanslag får finnas i namnet.

Exempel på en fil /usr/lib/bup/volymnamn:

```
Före backup:    d:  daytape1,   daytape2,   daytape3
                w:  weektape1,  weektape2,  weektape3
                m:  monthtape1, monthtape2, monthtape3

Efter första    d:  daytape1,   daytape2,   daytape3
vecko-kopier-   w:  weektape2,  weektape3,  weektape1
ingen:          m:  monthtape1, monthtape2, monthtape3
```

bup spar filerna på kopieringsmediet i överensstämmelse med följande tar-kommandos tillval och listan med filer lägges till loggfilen tillsammans med volymnamnet.

```
tar -cfbk device blocksize volumesize  files .....
```

I loggfilen /usr/lib/bup/log är varje fil, som är lagrad som en kopieringsvolym, listad på en rad med volymnamn, det fulla sökvägsnamnet och tidpunkt för kopieringen. Endast den senaste tidsangivelsen för kopiering sparas i loggfilen. Exempel på en loggfil.

```
w week2  Mar 17 13:40 1987 /usr/manu/dx/bup.doc
w week3  Feb 17 14:56 1987 /usr/manu/dx/login.doc
m monthl Mar 15 16:31 1987 /usr/manu/dx/mail.doc
```

OPTIONER

-V Skriver **bup**-kommandots versionsnummer och en kortfattad upplysning om användningen.

-v (Verbose=mångordig). I kombination med tillvalet -b eller -x skrivs all normal utdata från **tar** ut på standard output.

-m Ställer specificerade filer, som skall säkerhetskopieras, i kö då den månatliga kopieringen skall göras (med **bup** -bw).

-w Ställer specificerade filer, som skall säkerhetskopieras, i kö då veckans kopiering skall göras (med **bup** -bw).

-d Ställer specificerade filer, som skall säkerhetskopieras, i kö då den dagliga kopieringen skall göras (med **bup** -bd).

-a Ställer specificerade filer, som skall säkerhetskopieras och sedan raderas, i kö , då kopiering för arkivet görs (med **bup** -ba). I detta fall raderas även motsvarande rad i bup:s kommandofil då arkivkopieringen görs.

-k Flyttar en tidigare köordnad fil från bup:s kommandofilen. Namnet på filen/biblioteket skall återges på exakt samma sätt som uppgivits i kommandofilen, inklusive ev. specialtecken.

-bm Utför den månatliga säkerhetskopieringen. Alla filer som har köordnats med '-m', '-w' eller '-d'-tillval kommer att kopieras på den enhet som angetts.

-bw Utför veckokopieringen. Alla filer som har köordnats med '-w' eller '-d'-tillval kommer att kopieras på den enhet som angetts.

-bd Utför den dagliga kopieringen. Alla filer som har köordnats med '-d'-tillvalet kommer att kopieras på den enhet som angetts.

-ba Utför arkivkopiering till den angivna enheten och raderar filerna. Detta påverkar enbart filer som är köordnade med tillvalet '-a'. Motsvarande -a-rader i bup:s kommandofil raderas automatiskt efter det att säkerhetskopieringen är genomförd.

-x Hämtar filer från ett kopieringsmedium på den givna enheten. **bup** ber användaren att skriva in det riktiga volymnamnet och hämtar sedan de filnamn som är givna som argument. Om fil- eller biblioteksnamnet inte är givet med det fullständiga sökvägsnamnet så lägger **bup** till det gällande bibliotekets fullständiga sökvägsnamn. Filnamnen skall anges på samma sätt som de är listade i loggfilen. **bup** sparar alltid filerna med det fullständiga sökvägsnamnet. Om användaren vill lägga in en fil i ett annat bibliotek för att där kunna undersöka den, kan **tar**-kommandot med tillval -A användas istället för **bup** -x.

-u Specificerar volymnamn, som vanligtvis är ett bandnummer eller liknande. Detta namn lagras i loggfilen, och vi rekommenderar därför att bandet/disketten märks på samma sätt. Om -u tillvalet inte är specificerat och något av tillvalen '-bm', '-bw' eller '-bd' är använt kommer **bup** att försöka hämta det riktiga volymnamnet från filen /usr/lib/bup/volymnamn.

-l Listar filerna från bup:s loggfil. -l kan kombineras med -u då man vill välja ut filer enbart från ett specificerat volymnamn. Om filnamn är specificerade som argument kommer endast information om dessa att listas. Observera att specialtecken (wildcards) inte är tillåtna och att man måste ange det fullständiga sökvägsnamnet.

-r Rekursiva parametrar till **find**. Detta tillval gör att alla följande parametrar på kommandoraden behandlas som parametrar till kommandot **find**. Parametrarna lagras i bup:s kommandofil och används av **find** då själva säkerhetskopieringen råkar på filerna. Ett mellanslag måste skilja -r och parametrarna åt. Observera att inga '-exec' eller '-print'-parametrar kan specificeras. Då -r används kan inga normala filnamn specificeras.

-nm Visar (default) volymnamn som skall användas vid nästa månatliga säkerhetskopiering. Namnet hämtas från filen /usr/lib/bup/volymnamn.

-nw Visar (default) volymnamn som skall användas vid nästa veckas säkerhetskopiering. Namnet hämtas från filen /usr/lib/bup/volymnamn.

-nd Visar (default) volymnamn som skall användas vid nästa dags säkerhetskopiering. Namnet hämts från filen /usr/lib/ bup/volymnamn.

FILER

/usr/lib/bup/cmd	Bup:s kommandofil med filnamn och find-argument.
/usr/lib/bup/log	Loggfil innehållande volymnamn och tidpunkt för säkerhetskopieringen
/usr/lib/bup/volumename	Fil innehållande (default) volymnamn för den månatliga, veckovisa och dagliga säkerhetskopieringen.

EXEMPEL

```
bup -w '*'
```

Detta kommando köordnar alla filer i det gällande biblioteket som skall kopieras vid nästa veckas säkerhetskopiering.

```
bup -m -r -name Ö*.c  -mtime -30
```

Detta kommando köordnar all *.c-filer i det gällande biblioteket som har ändrats under de sista 30 dagarna och som skall kopieras vid nästa månatliga säkerhetskopiering.

```
bup -bw -utape_123 /dev/st0
```

Detta kommando utför veckans säkerhetskopiering på ett band med volymnamnet 'tape_123'. Om tillvalet -u inte hade givits skulle volymnamnet vara det första namnet på raden märkt w: i filen /usr/lib/bup/volymnamn.

```
bup -ba -v /dev/sf0
```

Detta kommando kopierar alla filer som först köordnats med **bup** -a och sedan raderats ur systemet. Eftersom tillvalet -v har givits, skrivs all **tar**-info ut under kopieringen.

```
bup -x /dev/st0 file1 file2 file3
```

Detta kommando återladdar file file1 file2 och file3 it det gällande biblioteket. **Bup** visar det riktiga volymnamnet som skall läggas in innan filerna laddas. Om filer som inte finns i det gällande biblioteket skall återladdas måste filernas fullständiga sökvägsnamn anges.

OBSERVERA

Säkerhetskopiering av specialfiler (t ex i /dev) kan inte utföras med **bup**. De kopieras med tillvalet -S i **tar**.

Filernas ändringstider visas inte i loggfilen; endast tidpunkten för säkerhetskopieringen visas.

12

12. Cron

12. Cron

cron är en administrativ demon-process som startar upp program vid specificerade tidpunkter. I system som baserades på tidigare UNIX-versioner specificerades alla program som skulle exekveras i en enda fil (**crontab**), vilken normalt hade speciella åtkomsttillstånd för att kunna hantera systemadministrativa processer. För att använda dessa funktioner krävdes särskilda tillstånd.

I D-NIX hanterar **cron** både administrativa filer och användarnas egna crontab-filer och kommandot **crontab** används av vanliga användare för att hantera dessa filer.

crontab hanterar användarnas crontab-filer. Programmet testar tillstånd och kopierar användarens filer och kommandon till rätt bibliotek för att **cron**-demonprocessen ska kunna utföra dem.

cron hanterar även begäran från kommandona **at**(1) och **batch**(1) i utvecklingssystemet. Kapitel 12.3 ger en kort beskrivning av detta, i övrigt hänvisas till dokumentationen för utvecklingspaketet.

12.1 Administrativa filer

Detta kapitel beskriver filer som används av **cron** och som måste skapas och ändras av den systemansvarige. Alla dessa filer finns i biblioteket **/usr/lib/cron**.

Filer med kommandon som skall exekveras av **cron** samt crontab-tabeller finns i biblioteket **/usr/spool/cron**.

Nedan är en lista som visar vad dessa bibliotek kan innehålla, med de privilegier som normalt används.

```
-rwx------   bin   bin   /etc/cron
drwxr-xr-x   bin   bin   /usr/lib/cron
-rw-r--r--   bin   bin   /usr/lib/cron/cron.allow
-rw-r--r--   bin   bin   /usr/lib/cron/log
-rw-r--r--   bin   bin   /usr/lib/cron/queuedefs
drwxr-xr-x   bin   bin   /usr/spool/cron
drwxr-xr-x   bin   bin   /usr/spool/cron/crontabs
-r--r--r--   bin   bin   /usr/spool/cron/crontabs/adm
-r--r--r--   bin   bin   /usr/spool/cron/crontabs/root
-r--r--r--   bin   bin   /usr/spool/cron/crontabs/sys
-r--r--r--   bin   bin   /usr/spool/cron/crontabs/uucp
```

Om utvecklingssystemet ingår, finns även följande filer.

```
-rw-r--rr--  bin   bin   /usr/lib/cron/.proto
-rw-r--r--   bin   bin   /usr/lib/cron/at.allow
drwxr-xr-x   bin   bin   /usr/spool/cron/atjobs
```

Kommunikationen mellan användar-kommandon och **cron** sker genom namngivna pipes (FIFO-filer) i filsystemet.

12.1.1 Filerna allow eller deny

Olika användare kan tillåtas eller förbjudas att använda sig av kommandot **crontab** genom listor i filerna cron.allow eller cron.deny.

Filen **cron.allow** kan innehålla en lista över de användare (inloggningsnamnen) som får lägga kommandofiler i kö med **crontab**. Om filen cron.allow inte finns, kan istället en lista över de användare som inte får använda **crontab** finnas i filen **cron.deny**.

Om filen cron.allow existerar, ignoreras filen cron.deny helt. Om varken cron.allow eller cron.deny existerar är det endast root (med superuserprivilegier) som kan använda kommandot **crontab**.

En sammanfattande tabell visar tillstånden att använda kommandona:

Filen 'allow'	Filen 'deny'	Användare med tillstånd
saknas	saknas	Endast root
saknas	tom	Alla användare
saknas	finns	Hindrar listade användare
tom	(ignoreras)	Endast root
finns	(ignoreras)	Tillåter listade användare

12.1.2 Köer för cron

cron kan dynamiskt begränsa antalet processer som exekveras samtidigt från köerna. Den klarar av upp till 26 olika köer och begränsar antalet processer ur varje kö separat. I filen **queuedefs** ska finnas definitioner av de köer som används.

OBS! I grundpaketet används endast den kö som kallas 'c' med kommandot **crontab**, och om denna kö inte definierats i queuedefs används standardvärden istället.

Varje rad i filen queuedefs skall ha följande format:

```
q.NNjNNnNNw
```

där:

q är en bokstav a-z som är kön namn.

NNj är max antal processer ur denna kö som får exekveras samtidigt. NN är ett heltal och standardvärdet är 100 om det ej anges.

NNn är argumentet till nice-kommandot som används när en process startas från kön. Standardvärdet är 2 om det ej anges.

NNw är tiden (i sekunder) innan cron försöker starta en process ur denna kö igen efter att villkoren visat sig inte uppfyllda. Standardvärdet är 60 (1 minut) om det inte anges.

Ett tomt fält sätts automatiskt till standardvärdet. Följande är ett exempel på en **queuedefs**-fil:

```
a.4jln
b.2j2n90w
c.0n10j
n.120w4n1j
```

Tredje raden beskriver kön 'c', som används för kommandot **crontab**. Raden anger att max tio processer ur kön får köras samtidigt och att nice-värdet är 0. Om kön innehåller fler än tio begäran kommer alla utom de tio första att få vänta. Eftersom ingen väntetid definierats här kommer **cron** att testa varje minut (standardtiden 60 sek) och starta den elfte processen ur denna kö då någon av de första tio blivit klar. Med nice-värdet 0 kommer processerna att exekveras med samma prioritet som normala kommandon. Med ett högre nice-värde kan de exekveras med en lägre prioritet.

De andra raderna i exemplet beskriver andra köer, som endast används med **at**(1) och **batch**(1) som inte ingår i grundsystemet. Till exempel anges att bara en process får köras i kö 'n', (aktiverad av **at**), varvid systemet väntar minst 120 sekunder mellan testerna om nästa kommando i kön ska få exekvera. Processen körs med nice-värdet 4.

Om kö-definitionsfilen ändras läser **cron** den innan nästa process ur kön startas. Om filen inte existerar används standardvärdena.

12.1.3 Log-fil

cron loggar alla kommandon som startas, avslutas och deras status i filen **log**. Poster som inleds med tecknet '>' mostvarar start av ett kommando. Två poster skrivs för varje kommando. Först lagras kommandot och sedan kommer en post med login-namn, process-id, könamn och tiden när kommandot startades. Poster som inleds med tecknet '<' motsvarar terminering av ett kommando och liknar start-poster, med tillägget att kommandots utgångsstatus också lagras. Poster som inleds med tecknet '!' innehåller statusinformation.

12.1.4 Format hos crontab-filerna

Med den nya **cron** processen har varje användare en egen individuell fil för crontab. Dessa filer ligger i biblioteket /usr/spool/cron/crontabs. Namnet på crontab-filen används som användarnamn vid exekveringen för att erhålla rätt filåtkomsttillstånd för användare och grupp.

Nedan är ett exempel på en rad i en crontab-fil med namnet sys. Den exekveras då med alla tillstånd som användare sys har.

```
0 19-7 * * * /usr/lib/sa/sal >/dev/null &
```

12.2 Allmänt om funktionen hos cron

Kommandot **crontab** skapar kommandon till **cron** och sänder dessa genom en spciell fil /usr/lib/cron/FIFO. När **crontab** gör en ny crontab-fil eller ändrar eller tar bort en redan befintlig, sänds ett meddelande via FIFO som gör att **cron** läser av filen och lägger in exekveringsbegäran i sin tidskö.

Eftersom **cron** läser alla filer i /usr/spool/cron/crontabs varje gång **cron** startas kommer exekveringsbegäran i köerna att aktiveras även om systemet stängs av och startas upp igen. Dessutom är det även möjligt att manuellt lägga in crontab-filer, förutsatt att användaren har lämpliga skrivtillstånd, varvid även dessa läses in nästa gång **cron** startas. Därvid används inte cron.allow eller cron.deny, vilka enbart styr **crontab**-kommandot. **Cron** startas normalt på nytt då datorsystemet startas upp eller vid övergång till fleranvändarmod i operativsystemet. Därför bör enbart super-user ha skrivtillstånd i dessa bibliotek.

cron-processen försöker rapportera fel som orsakar att den stoppas genom felmeddelande till systemkonsolen. Varningsmeddelanden ges även vid användning av **crontab** om **cron** inte är aktiverad i systemet.

12.3 Användning av cron med utvecklingspaketet

I utvecklingspaketet som beställs separat, finns två ytterligare kommandon som sänder exekveringsbegäran till **cron**. **Cron** hanterar dessa på liknande sätt som från **crontab**. Nedan listas de filer som då används. För detaljer, se handböckerna för utvecklingssystemet.

Med **batch**(1) används kön med namnet 'b' i filen queuedefs, medan kommandot **at**(1) kan använda vilken kö som helst. De ytterligare filer som då finns är:

/usr/spool/cron/atjobs/*	Bibliotek med kommandofiler som väntar på exekvering.
/usr/lib/cron/at.allow	Motsvarande cron.allow, men för **at** och **batch**.
/usr/lib/cron/at.allow	Motsvarande cron.deny.
/usr/lib/cron/.proto	Prototypfil för att skapa **at**-filer.
/usr/lib/cron/.proto.q	Separata prototypfiler kan användas för olika köer.

Prototypfiler används för att standardisera informationen som skrivs till de kommandofiler som skapas med **at**. Om en fil med namnet **.proto.q** finns, där 'q' anger den kö den skall användas till, kopieras denna fil till den skapade kommandofilen. Annars används den allmänna prototypfilen **.proto**. Kommandot **at** avbryts med ett felmeddelande om inte prototypfil existerar.

Följande shellvariabler kan användas i prototypfilerna och substitueras då en kommandofil skapas av **at**.

%m användarens nuvarande tillståndsmask vid skapande av filer (se **umask**).

%l användarens nuvarande max filstorlek vid skapande av filer (se **ulimit**(2)). Det är bokstaven lilla l som används som variabelnamn.

%d namnet på nuvarande aktuellt bibliotek.

%t tiden då processen skall exekveras. Anges som ett numeriskt värde i sekunder.

%< standard input läses härifrån och tills filslut och lägges in i kommandofilen.

Nedan är ett exempel på en prototypfil. Dessa kommandon lägges in i den kommandofil som skapas, före användarkommandona, men efter att normala standardvariabler definierats.

```
cd ¤d
ulimit ¤l
umask ¤m
¤<
```

13

14

15

15. Yttre enheter

15. Yttre enheter

Alla yttre enheter som ska anslutas till systemet måste uppfylla två kriterier. Det första är att de är mekaniskt/elektriskt anslutna till systemet. Det andra är att de måste kopplas samman logiskt med systemet operativsystemet via drivrutiner.

I grundutförande levereras systemet med fyra seriella portar, ett skivminne av winchestertyp, en 5 1/4 tums diskettenhet, eventuellt kassettbandstreamer samt möjlighet att ansluta följande expansionsenheter:

Dator	Interna seriella portar	Interna DataBoard kort	Interna VME-kort	Yttre winchester minnen **)
DS90-10	2 * 4 st	4 st	2 st	7 st
DS90-10E	2 * 4 st	---	---	7 st
DS90-11	2 * 4 st	3 st	---	7 st
DS90-20	2 * 4 st	4 st	3 st	6 st
DS90-21	2 * 4 st	3 st	6 st	6 st

**) Av de yttre SASI-enheterna kan någon vara en yttre streamerbandstation istället för ett skivminne.

* Två interna terminalexpansionkort med vardera fyra extra terminalanslutningar. Dessa monteras i direkt anslutning till datorkortet.

* Tre eller fyra I/O-kort ur DataBoardserien kan sättas in i den interna expansionsracken, för t ex nät-anslutning, modemanslutning, yttre diskettenhet, ytterligare terminalanslutningar m m.

* Två eller flera VME-kort kan sättas in i den interna expansionsracken. Här kan ett större antal extra terminalanslutningar erhållas via terminalkoncentratorkort.

* Yttre winchesterenheter ansluts till en kontakt på baksidan. Antalet enheter i tabellen ovan förutsätter att en intern kassettstreamer finns i systemet. I annat fall kan ytterligare en enhet anslutas.

I detta kapitel presenteras först en lista över de logiska kopplingarna för alla i systemet förekommande yttre enheter. Observera att listan även innehåller logiska enheter för vissa interna drivrutiner, som t ex primärminnet.

I systemet benämns de logiska kopplingarna i två nivåer:

* **Major device : Typ av enhet**
* **Minor device : Ordningsföljden i denna typ av enhet.**

De svenska beteckningarna för dessa nivåer är:

Major device : Primärt enhetsnummer
Minor device : Sekundärt enhetsnummer

15.1 Major och minor device

Alla yttre enheter som finns anslutna finns i biblioteket /dev. Ett exempel på hur några yttre enheter kan vara representerade visas nedan, vilket är några rader ur det som visas med kommandot **ls**.

```
ls -l /dev.

crw--w--w- 1 bc    other    1,   1 Dec 17  1985 console
brw-rw-rw- 1 root  root     5,128 Sep 14 12:17 st0
crw-rw-rw- 1 root  root     1,   5 Sep  1  1985 tty05
```

Det första tecknet visar av vilken typ den anslutna enheten är, de följande nio tecknen visar vilka som får använda enheten och hur. Det första tecknet är ett **b** om enheten är blockorienterad, t ex bandstationer, och **c** om den är teckenorienterad d v s sänder ett tecken i taget.

Det finns dessutom två tal som anger hur enheten är ansluten, fysiskt och logiskt. De talen går under benämningen primärt enhetsnummer (major device) och sekundärt enhetsnummer (minor device).

Den primära enhetsnumret beskriver i princip vilken drivrutin som ska användas för respektive enhet. Det sekundära enhetsnumret beskriver vilken fysisk anslutning under denna drivrutin som används, t ex drivenhet eller linjenummer, men kan även innehålla annan information, som t ex diskettformat och liknande.

En tabell över de vanligaste primära och sekundära enhetsnumren finns på nästa sida.

Notera att dessa enhetsnummer och beteckningar kan skilja sig mellan olika maskinvarukonfigurationer.

Även mellan olika systemmodeller skiljer sig parametrarna för winchesterskivminnena, kassettstreamer och diskettenheterna.

Primär Major	Sekundär Minor	Namn	Förklaring
1			Enheten styrs av drivrutinen för de seriella portarna.
	0	lp	Skrivarporten
	1	console	Porten för huvudkonsolen
	1	systty	(Porten för huvudkonsolen)
	1	syscon	Virtuell huvudkonsol(Enanvändarmod)
	2	tty02	tty02
	.	.	.
	.	.	.
	11	ttyll	ttyll
2	0	tty	Enhet för den terminal användaren är inloggad på.
3			Enheten styrs av den drivrutin som hanterar den interna maskinvaran.
	0	mem	Primärminnet
	1	kmem	kmem
	2	null	Null
	3	inoutb	Rutin för I/O på bytenivå.
	4	inout	Rutin för I/O på ordnivå.
	5	autosw	Läser nyckel-omkopplaren
	6	bclock	Systemklockan
	7	nvram	Startparametrar, non-voilatile RAM.
	8	DBinoutb	DataBoard I/O
	9	error	Log för systemmeddelanden
5			Enheten styrs av drivrutinen för winchesterenheter och streamer. Olika sekundära enhetsnummer används i olika system enligt tabellen nedan.

Datormodellerna DS90-10/-10E/-20/-21:

16	si0	SCSI 1, id 0:	(Intern winchester)
128	st0	SCSI 0, id 0:	(Intern kassett-streamer)
129	nst0		(Icke återspolande intern streamer)
2	si2	SCSI 0, id 1:	(Yttre winchester)

Datormodell DS90-11:

0	si0	SCSI 0, id 0:	(Intern winchester)
144	st0	SCSI 1, id 0:	(Intern kassett-streamer)
145	nst0		(Icke återspolande intern streamer)
18	si2	SCSI 1, id 1:	(Yttre winchester)

Primär Major	Sekundär Minor	Namn	Förklaring
6			Enheten styrs av drivrutinen för bandstationer.
7			Drivrutin för SP1/Centronics gränssnitt.
	x		Se avsnitt 5.6.2
8			Drivrutiner för seriella kanaler via 4117/4118 DataBoard-kort.
	x		Se avsnitt 5.6.6 - 5.6.7
9			Enheten styrs av drivrutinen för diskettenheterna.
	8	mf0	5 1/4 tums intern diskettenhet. Se kapitel 5.3.
10			Om ett kommunikationskort ansluts (4004) används en drivrutin som har enhetsnummer 10.
11			Drivrutin för terminalkoncentrator på ett eller flera VME-kort.
	0	tty12	Serieportar: Minor: 0..9 på kort 0
	1	tty13	16..25 på kort 1
	...		32..45 på kort 2
			Enhetsnamnen kan väljas i anslutning till de interna tty-portarna.
	128	tc0	Styrning/status på kort 0
	144	tc1	Styrning/status på kort 1
	160	tc2	Styrning/status på kort 2
12			Drivrutin för VME-bushantering.
	0	VMEdrv	Rutin som sänder kommandon och väntar på avbrottssignal.
13			Drivrutin för avbrottshantering.
	0	DBIdrv	Avbrottshantering DataBoard
14			Reserverad
15			Reserverad för GPIB.
16			Hantering av grafikkort.

15.2 Aktivering av drivrutiner till yttre enheter

Alla yttre enheter som har blivit mekaniskt anslutna används via biblioteket /dev. Men för att det ska vara möjligt måste drivrutinerna aktiveras i systemet. De drivrutiner som kan aktiveras är de som från början finns inlagda i operativsystemet. För att aktivera dessa använder man sig av kommandot **mknod**.

mknod används t ex på följande vis:

```
/etc/mknod /dev/tty08 c 1 8
```

I det här exemplet definierar vi till en terminal med namnet tty08. Enheten arbetar med tecken (c) och drivrutinen för seriell överföring ska användas (1) och den ansluts till I/O-kanal 8.

mknod skapar en specialfil i biblioteket /dev. Första argumentet är filens namn. Det andra argumentet är b om specialfilen är av blocktyp (diskar, tape) eller c om den är av charactertyp, d v s sänder ett tecken i taget. De sista argumenten är tal som anger primärt enhetsnummer (major device) och sekundärt enhetsnummer (minor device t ex unit, drive eller linjenummer) och kan vara både decimala och oktala.

Tilldelningen av enhetsnummer är specifik för varje system enligt föregående kapitel.

OBS! Om extra enheter utöver grundsystemet är aktiverade i systemet bör systemadministratören inkludera dessa vid en totalbackup av systemet, för den händelse att en total omgenerering av det skivminne, som systemet har som root device, måste göras. Vid en installation av systemet från början aktiveras de i grundsystemet befintliga enheterna av installationsprogrammet **harddoit** på den medföljande mount-disketten. Om sekundära eller primära enhetsnummer har ändrats för sådana enheter, måste dessa därför ändras med **mknod** efter omgenerering av systemet.

15.3 Interna diskettenheten - /dev/mf0

Den interna diskettenheten är en 5 1/4 tums diskettenhet och kan hantera disketter med 80 spår och en lagringskapacitet om 720 kbytes samt 1.2 Mbytes.

Diskettenheten nås genom enheten **/dev/mf0**.

Den är ansluten logiskt till systemet via det primära enhetsnumret **9** (major device) det sekundära enhetsnumret **8** (minor device).

Vid systemstart från disketten via systemets bootstrap PROM anges kanalnummer 8 genom enhetsnamnet **mf(8,0)** för 1.2 Mb disketter. Alternativt kan kanalnummer 64 anges, dvs **mf(64,0)** vid start från en 720 kb diskett.

För bästa resultat och största säkerhet i datalagringen rekomenderas dubbelsidiga disketter avsedda för lagring med dubbel packningstäthet på 80 spår.

För 1.2 Mbytes lagringskapacitet krävs att "high-density"-disketter används,

På följande sidor finns en utförlig beskrivning av olika diskettformat som kan hanteras av systemet.

15.3.1 Diskettformat, allmänt

Disketter har många olika lagringsformat. Intentionen med programvaran i systemet är att vara så flexibel som möjligt med avseende på olika sektorstorlekar, steghastigheter, olika sektorer/spår, enkel/dubbel/hög densitet, enkel/dubbelsidiga ..... etc. Nedan följer en kort beskrivning av de format och parametrar som systemet klarar.

Diskettdrivrutinen är utformad för att hantera 5 1/4 tums mini- och 8 tums standarddisketter med dubbel densitet (DD) och dubbelsidig (DS) lagring. En 8 tums standard diskett har 77 spår. 5 1/4 tums mini disketter är normalt av typen dubbelspårs (DT) med 80 spår.

Drivrutinen klarar även av 8 tums disketter med enkel densitet om disketten har 128 bytes/sektor.

Det är möjligt att sätta in en enkelspårs 5 1/4 tums diskett i en dubbelspårs drivenhet. Den här konfigurationen kan endast användas när 'sense' biten är satt i sekundära enhetsnumret och disketten kan enbart läsas, inte skrivas eller formatteras. Om det är nödvändigt att skriva enkelspårs disketter så måste en enkelspårs drivenhet kopplas in.

Allt som drivrutinen gör baseras på det sekundära enhetsnumret (minor number) vilket består av 8 bitar. Den betydelsefullaste biten är 'sense' biten (bit 3) och om den är satt kommer drivrutinen att försöka sätta upp diskettparametrarna själv, men om inte kommer de att tas från minornumret. Följande parametrar kan automatiskt sättas av drivrutinen, sektorstorlek, enkel/dubbelsidig, enkel/dubbelspår och antal sektorer/spår.

Det finns åtminstone två sökmetoder för dubbelspårs disketter, alternerande och sekvensiell. Alternerande fungerar så att för varje nytt spår så söks sida 0 och efter det sida 1. Sekvensiell fungerar så att hela sida 0 söks först innan sida 1 söks. Normalt använder drivrutinen alternerande sökning men undantag är 5 1/4 tums diskett och 256 bytes/sektor format där sekvensiell sökning används.

Drivrutinen klarar även de nya "high density" 5 1/4 tums disketterna (1.2 Mbyte). Dessa disketter kräver att 'sense'-biten är satt och att diskettenheten kan hantera denna typ av disketter.

Minor-numret är följande om 'sense' biten (bit 3) är satt.

Bit 7 = 1 för 8 tums diskett
= 0 för 5 1/4 tums diskett.

Bit 6-5 Steghastighetsvärde, andvänds vid sökning. Detta ger användaren möjligheten att specificera andra steghastigheter än standard som är 0. För andra värden hänvisas till manualen för Western Digital Floppy Controller 179x. Generellt kan sägas att högre värden ger längre stegtid.

Bit 4 Reserverad, bör vara 0.

Bit 3 Sense bit = 1.

Bit 2 Reserverad, bör vara 0. Temporärt anger bit 2 = 1 att valda diskparametrar visas på skärmen då disketten läses, men kan användas till annat i framtiden. Se kapitel 15.3.2.

Bit 1,0 Fysisk enhet 0..3.

Minor bitar om 'sense' är 0.

Bit 7 = 1 för 8 tums diskett
= 0 för 5 1/4 tums diskett.

Bit 6,5 Sektorstorlek. 0x00 = 256 byte/sektor.
0x20 = 128 byte/sektor.
0x40 = 512 byte/sektor.
0x60 = 1024 byte/sektor.

Bit 4 Enkelsidig diskett. Endast sida 0 söks.

Bit 3 Sense bit = 0.

Bit 2 Antal sektorer/spår

Bit-värde:		0	1	
128	sf	26	25	sekt/spår
256	sf	26	25	
512	sf	16	15	
1024	sf	8	7	
128	mf	16	15	
256	mf	16	15	
512	mf	9	8	
1024	mf	5	4	

Bit 1,0 Fysisk enhet 0..3.

Används sekundära enhetsnumret (minor) 128+32+16+2 kommer drivrutinen att läsa enkel densitets 8 tums standarddiskett på enhet 0. Detta är det enda sättet att läsa 8 tums disketter med enkel densitet. Sense optionen är inte implementerad för enkel densitet

Rekommenderade sekundära enhetsnummer (minor) för DS90.

```
Mini diskett enhet 0     8    Sense och enhet 0.
Mini diskett enhet 1     9    Sense och enhet 1.
```

Rekommenderade parametrar vid formattering av disketter.

Några exempel på kombinationer ges till höger med följande förkortningar:

```
DD = Dubbel densitet
ST = Enkelspår (40 spår)
DT = Dubbelspår (80 spår)
HDENS = High density disketter (80 spår,DD,DT)
NCR = NCR-kompatibla disketter (80 spår,DD DT)
```

	Nhead		
Dubbel sida	2		
Enkel sida	1		
	Ncyl (Nspår)		
Standard 8 tum std	77		
Dubbelspår(DT) 5 1/4 tum	80		
Dubbelspår(DT) 5 1/4 tum (HDENS)	80		
Enkelspår (ST) 5 1/4 tum	40		
	Nsek/spår	**Exempel**	
Enkel densitet 8 tum std:			
128 bytes/sek	26	Std	256 kb
Dubbel densitet 8 tum std:			
256 bytes/sek	26	Std	1 Mbyte
512 bytes/sek	16		
1024 bytes/sek	8		
Dubbel densitet 5 1/4 tum mini:			
256 bytes/sek	16	DD ST	320 kb
		DD DT	640 kb
512 bytes/sek	9	DD ST	360 kb
	9	DD DT	720 kb
	8	NCR	640 kb
	15	HDENS	1.2 Mbyte
1024 bytes/sek	5		

15.3.2 Speciella diskettformat

Disketter med D-NIX filsystem initieras med **/etc/mkfs** enligt kapitel 15.2 och hanteras direkt av operativsystemet.

Disketter med MS-DOS-format kan hanteras med egna filhanterare, MS-DOS filhanterare, som kan köpas till systemet. Då kan de flesta vanliga kommandon användas för att läsa/skriva data, lista filer, skapa och ta bort filer.

Disketter med format enligt SBC-DOS/ABC-DOS kan hanteras med egna filhanterare, ABC-filhanterare som kan köpas till systemet. Då kan de flesta vanliga kommandon användas för att läsa/skriva data, lista filer, skapa och ta bort filer.

På disketter utan filsystem kan filer lagras med kommandot **cpio** eller med **tar**, vilka är standard i alla Unix-system.

I alla dessa fall måste disketten vara formatterad. Om utbyte av disketter mellan olika datorer skall ske måste formatteringen överensstämma.

Diskettformat för en okänd diskett visas på bildskärmen vid läsning om den läses via en enhet som definierats (/etc/mknod) med både 'sense'-biten och bit 2 satt i sekundära enhetsnumret. Exempelvis kan följande kommandon användas för att läsa formatet på en 5 1/4 tums diskett.

```
/etc/mknod  /dev/mf00 c 9 12
dd if=/dev/mf00 of=/dev/null bs=1k
```

OBS!! Denna funktion (bit 2 =1) med 'sense' i minor-numret är temporär och kan eventuellt användas till annat i framtiden.

Exempel på andra datorers diskettformat är, med beteckningarna i tabellen i föregående avsnitt:

```
DD DT   360 K    IBM-PC
HDENS   1.2 Mb   IBM-AT
NCR     640 K    NCR XD, NCR TOWER-32, SPERRY 5000/20-50
```

15.4 Interna skivminnesenheten - /dev/si0

Det fasta skivminnet har enhetsnamnet **/dev/si0**.

Denna enhet är aktiverad vid leverans och används av filhanteraren i systemet.

Enheten har det primära enhetsnumret **5** (major device). Det sekundära enhetsnumret (minor device) varierar mellan datormodellerna eftersom olika SASI/SCSI-portar används.

Vid systemstart med uttryckligt angivande av enhet via systemets bootstrap PROM anges kanalnummer genom enhetsnamnet **si(X,0)**, där X är det sekundära enhetsnumret. Se nedan.

System	Sekundärt enhetsnummer	Enhetsnamn i laddnings-programmet	Enhetsnamn i D-NIX
DS90-11	**0**	si(0,0)	/dev/si0
DS90-10	**16**	si(16,0)	/dev/si0
DS90-10E	**16**	si(16,0)	/dev/si0
DS90-20	**16**	si(16,0)	/dev/si0
DS90-21	**16**	si(16,0)	/dev/si0

15.5 Intern kassettstreamer - /dev/st0

15.5.1 Intern kassettstreamer

En kassettstreamer finns monterad i frontpanelen och har enhetsnamnet **/dev/st0**. För vissa modeller är kassettstreamer en option.

Kassettbandet skall vara av typen 3M-kassetter för 10000 bpi. Lagringsformatet är enligt QIC-24 standarden.

Drivrutinen för kassettstreamern har det primära enhetsnumret **5** (major device), eftersom den styrs av samma drivrutin som winchesterenheten. Det sekundära enhetsnumret (minor device) varierar mellan datormodellerna. Se tabellen nedan.

En speciell drivrutin finns för kassettstreamern om man inte vill att bandet ska återspolas efter läsning. Enhetsnamnet är då **/dev/nst0** och det sekundära enhetsnumret (minor device) är enligt nedan.

OBS! Skrivning till kassettstreamern kan bara ske från början av bandet. Därför kan inte optionerna 'r' och 'u' användas i **tar** och bandet måste alltid återspolas före skrivning. Läsning kan däremot göras från valfri position.

/dev/nst0 skapas inte av **harddoit** från BOOT-disketten, utan måste aktiveras av användaren om den behövs.

Sekundärt enhetsnummer för kassettstreamer

System	Normal	Icke återspolande
DS90-00	12	0
DS90-11	**144**	**145**
DS90-10	**128**	**129**
DS90-10E	**128**	**129**
DS90-20	**128**	**129**
DS90-21	**128**	**129**

15.5.2 Kassettstreamer NCR-format

Med den interna kassettstreamerstationen kan exempelvis kassetter utbytas med NCR-datorer. Då måste emellertid noteras att data lagras på kassetterna i NCR med byte-swap.

Vid hantering av kassetter i format enligt kommandot **cpio** används då optionen **s** i D-NIX för att erhålla byte-swap.

Vid läsning och skrivning med kommandot **tar** måste data filtreras genom kommandot **dd** för att erhålla byte-swap enligt nedanstående exempel.

Läsning av NCR-kassett med **tar**:

```
dd if=/dev/st0 conv=swab  ö  tar xvf -
```

Skrivning av biblioteket mydir till NCR-kassett med **tar**:

```
tar cvfbk - 200 60300 mydir ö dd conv=swab of=/dev/st0
```

Tecknet ö är pipe-symbolen i shell.

15.6 Terminalkoncentrator VME/TC1

Drivrutiner finns för att expandera systemet med extra terminalanslutningar via VME-kort med vardera 10 anslutningar för de datormodeller där VME-kort kan anslutas. Antalet kort för de olika modellerna listas i början av kapitel 15.

(Ej DS90-11 eller DS90-10E)

Dessa anslutningar kan användas för anslutning av terminaler och skrivare eller för anslutning av kommunikationslinjer. Asynkron kommunikation används med upp till till 38400 Baud. Även split speed 75/1200 eller 1200/75 Baud kan användas.

Drivrutinen har primära enhetsnumret **11** (major device) och varje kort har tio portar med sekundära enhetsnummer (minor device) enligt tabellen nedan.

Enhetsnamnen (ttynn) är normalt enligt nedan och initieras vid installationen av medföljande installationsprogram.

För varje kort skall ett speciellt enhetsnamn (**/dev/tcn**) definieras genom vilket styrinfomation och status för kortet erhålles. De sekundära enhetsnumren (minor device) måste vara enligt tabellen för de olika korten.

Kort 0:	Enhetsnamn	Primärt enhetsnummer	Sekundärt enhetsnummer
	/dev/tc0	11	**128**
	/dev/tty12	11	**0**
	/dev/tty13	11	**1**
	...	...	...
	/dev/tty21	11	**9**

Kort 1:	Enhetsnamn	Primärt enhetsnummer	Sekundärt enhetsnummer
	/dev/tc1	11	**144** (=128+16)
	/dev/tty22	11	**16** (=0 +16)
	/dev/tty23	11	**17** (=1 +16)
	...	...	...
	/dev/tty31	11	**25** (=9 +16)

O s v med de sekundära enhetsnumren ökade med 16 för varje ytterligare kort.

15.6.1 Allmänt

Inledning

Terminalkoncentratorn VME/TC1 har tio (10) asynkrona tty-portar på ett VME-kort. Terminalkoncentratorn, nedan kallad TC, används till DS90-10 och DS90-20/21 datorerna och köps separat.

Terminaler ansluts via ett separat bakstycke till datorn med 10 eller fler standardkontakter av typ DA15P. Två flatkablar per TC-kort används för anslutning till bakstycket.

Maskinvara

Ett TC-kort består av ett intelligent seriekommunikationskort med Z80A CPU och kringkretsar. Kortet kommunicerar med huvuddatorn via ett tvåportsminne på kortet, vilket laddas med kommunikationsprogram från datorn. Mer detaljer finns i databladet för 4301-kortet.

Programvara

Programvaran för TC består av programmen tc4301, tcboot, tcset och tcpar4301. tc4301 är kommunikationsprogrammet för kortet som laddas med hjälp av programmet **tcboot**. Filen tcpar4301 innehåller parametrar som laddas tillsammans med tc4301 och kan ändras med kommandot **tcset**. Förutom dessa finns på leveransdisketten ett automatiskt installationskommando **install**, som skall användas vid installationen.

Vid installationen skall byglingar ställas in på kortet 4301 och enheter skall skapas i systemet, både för TC-kortet (tcn) och för alla 10 terminalportarna (ttynn). Detta görs med hjälp av **install**.

Leveransinnehåll

Följande komponenter levereras med varje VME/TC1.

```
4301-00    TC-kortet.
5186-00    Platta med 10 kontakter (DS90-10/20).
Två flatkabler mellan 4301 och 5186.
Datablad för 4301.
Diskett med programvara.
```

DS90-21 är redan försedd med platta.

15.6.2 Installation

Installation av maskinvaran

Innan kortet (korten) installeras, skall omkopplare SW1 och byglingarna S1...S4 ställas in samt eventuellt byglingarna för +/-12V enligt beskrivningen nedan. Olika adresser (SW1) skall användas för de olika korten om flera används.

Stäng av datorsystemet på normalt sätt så att spänningen kan kopplas bort. Första kortet (tc0) skall placeras i den vänstra positionen (nedre positionen i DS90-21), märkt "VME 1", närmast datorkortet i systemet. Det andra kortet (tc1) skall placeras i positionen "VME 2" och så vidare. Korten sättes in med komponentsidan åt höger (DS90-21: uppåt). För varje kort monteras en platta med kontakter i bakstycket och ansluts till respektive TC-kort med två flatkablar.

När detta är klart startas datorsystemet och programvaran för terminalkoncentratorn installeras enligt nedan, varefter terminalerna ansluts med standardkablar.

Installation av programvaran

En diskett innehåller programvaran för TC. Dessa filer kopieras till biblioteket /usr/lib/tc i systemet enligt informationen i medföljande Release Notice. Innan TC-systemet startas skall vissa enheter skapas. För vardera TC-kort skall en enhet för styrning skapas (/dev/tc0, /dev/tc1 och så vidare) och för varje tty-kanal skall en tty-enhet skapas. Tty-enheterna kan numreras som en fortsättning på de interna tty-portarna. Använd normalt de enhetsnamn (ttynn) som finns på etiketterna på kontakterna.

I DS90-21 system kan upp till sex TC-kort installeras, med upp till 60 extra tty-portar.

I DS90-20 system kan upp till tre TC-kort installeras, med upp till 30 extra tty-portar.

I DS90-10 system kan upp till två TC-kort installeras, med upp till 20 extra tty-portar.

Efter kopiering av programvaran till DS90 exekveras installationsprogrammet med följande kommando. **OBS!** att man måste logga in på systemet som super-user (root) för att kunna köra installationsprogrammet.

```
/usr/lib/tc/install
```

Detta program hjälper till vid installationen av VME/TC1 genom att ställa frågor och skapa enheter samt ändra i systemfiler för att automatiskt starta upp drivprogrammen för TC vid systemstart.

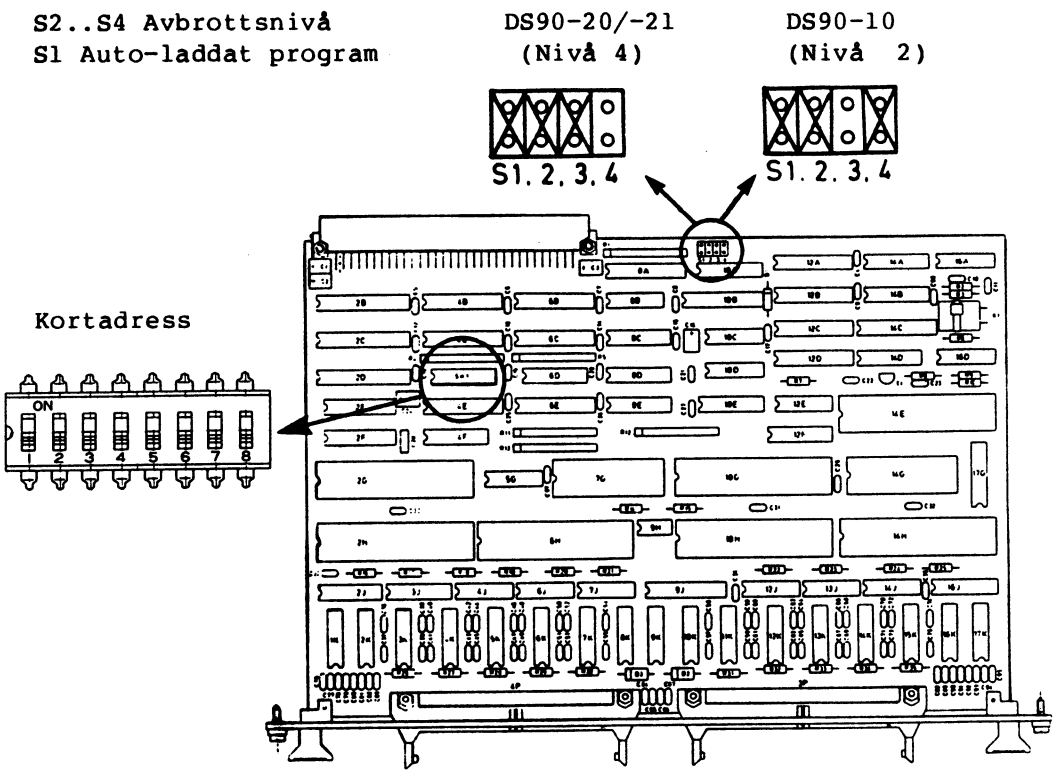
Byglingar

1 - Slut bygel S1, för att möjliggöra automatisk ladding av programmet till TC-kortet.

2 - Välj kortadresserna olika för de olika TC-korten med dipomkopplaren SW1 i position 4D (Se tabellen nedan).

3 - Välj avbrottsnivå, beroende på programvaran och datorsystemet (se figuren).

4 - Normalt lämnas alla byglingar öppna på vid kontakterna på plattan om inte externa anpassningsenheter används som kräver +12 eller -12V matningsspänning från datorn.



SW1 kortadresser för datorn DS90-21:

	1	2	3	4	5	6	7	8	Stift
1:a kortet (tc0)	C	C	C	C	C	C	C	C	C = Sluten/On
2:a kortet (tc1)	Op	C	C	C	C	C	C	C	Op = Öppen/Off
3:e kortet (tc2)	C	Op	C	C	C	C	C	C	
4:a kortet (tc3)	Op	Op	C	C	C	C	C	C	
5:a kortet (tc4)	C	C	Op	C	C	C	C	C	
6:e kortet (tc5)	Op	C	Op	C	C	C	C	C	

SW1 kortadresser för datorn DS90-20:

	1	2	3	4	5	6	7	8	Stift
1:a kortet (tc0)	C	C	C	C	C	C	C	C	C = Sluten/On
2:a kortet (tc1)	Op	C	C	C	C	C	C	C	Op = Öppen/Off
3:e kortet (tc2)	C	Op	C	C	C	C	C	C	

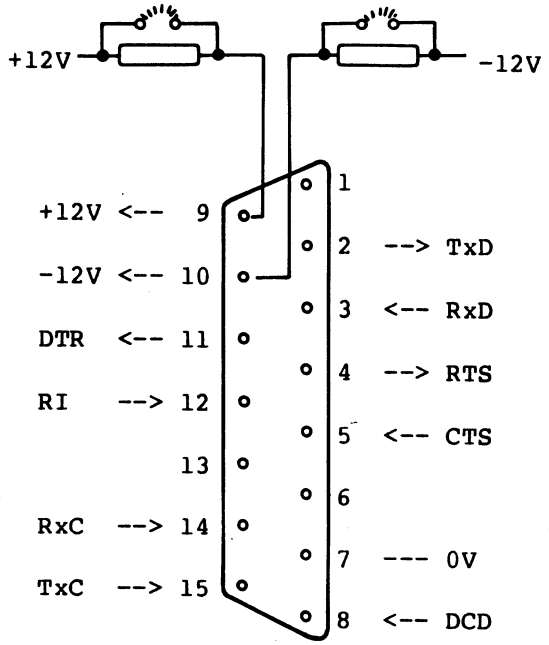
SW1 kortadresser för datorn DS90-10:

	1	2	3	4	5	6	7	8	Stift
1:a kortet (tc0)	C	C	C	Op	Op	Op	Op	C	C = Sluten/On
2:a kortet (tc1)	Op	C	C	Op	Op	Op	Op	C	Op = Öppen/Off

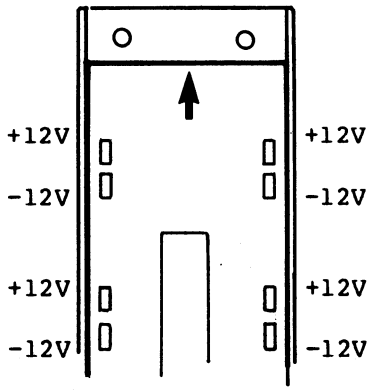
Seriekontakterna med byglingar

Terminalerna ansluts till varje TC-kort via en platta med tio standardkontakter av typ DA15P, med stiftplacering enligt nedan. Byglingar kan slutas om eventuella yttre adapterenheter kräver -12V eller +12V strömförsörjning från datorn. Normalt är alla dessa byglingar öppna och endast de logiska nivåerna +12V/-12V är tillgängliga i kontakten.

OBS! Maximala totala belastningen på +12V/-12V från datorn är begränsad. Se installationsmanualen för datorsystemet.



Byglingarna +12V/-12V finns i direkt anslutning till vardera kontakten, med +12V-byglingen ovanför -12V byglingen, då pilen på enheten pekar uppåt.



Byglingar för spänningsutgångar

Terminalkontakternas positioner på enheten är enligt figuren nedan. Motsvarande tty-enhetsnamn ges enligt standard för TC.

Kontakterna visas sedda utifrån baksidan av datorn.

För DS90-10 (2 kort) och DS90-20 (3 kort):

VME-1		VME-2		VME-3 (Endast Ds90-20)	
tty16	tty21	tty26	tty31	tty36	tty41
tty15	tty20	tty25	tty30	tty35	tty40
tty14	tty19	tty24	tty29	tty34	tty39
tty13	tty18	tty23	tty28	tty33	tty38
tty12	tty17	tty22	tty27	tty32	tty37

För DS90-21 (6 kort):

VME-6	tty71	tty70	tty69	tty68	tty67
	tty66	tty65	tty64	tty63	tty62
VME-5	tty61	tty60	tty59	tty58	tty57
	tty56	tty55	tty54	tty53	tty52
VME-4	tty51	tty50	tty49	tty48	tty47
	tty46	tty45	tty44	tty43	tty42
VME-3	tty41	tty40	tty39	tty38	tty37
	tty36	tty35	tty34	tty33	tty32
VME-2	tty31	tty30	tty29	tty28	tty27
	tty26	tty25	tty24	tty23	tty22
VME-1	tty21	tty20	tty19	tty18	tty17
	tty16	tty15	tty14	tty13	tty12

15.7 Andra yttre enheter

15.7.1 Anslutningar för övrig I/O

Anslutning av DataBoardkort

DataBoardkort kan anslutas till expansionskontakterna i systemet. Anslutningen sker med spänningen frånslagen och korten sättes in med komponentsidan åt höger (I DS90-21 och DS90-11 uppåt).

Detta gäller inte DS90-10E, som saknar expansionsrack.

Alla kontakter kan användas för DataBoard I/O-kort men endast en av kontakterna har extra signaler för anslutning av expansionskort (4203-00) till yttre expansionsrack. DataBoard-kort som inte använder avbrottssignaler kan placeras i valfri position, men vissa begränsningar gäller övriga kort enligt beskrivningen nedan för de olika datormodellerna.

I positionen för expansionskort finns extra signaler för hantering av externa interrupt och för anslutning av externa avbrottsavkodare (interrupt-scanner).

DataBoard-positioner i DS90-10:

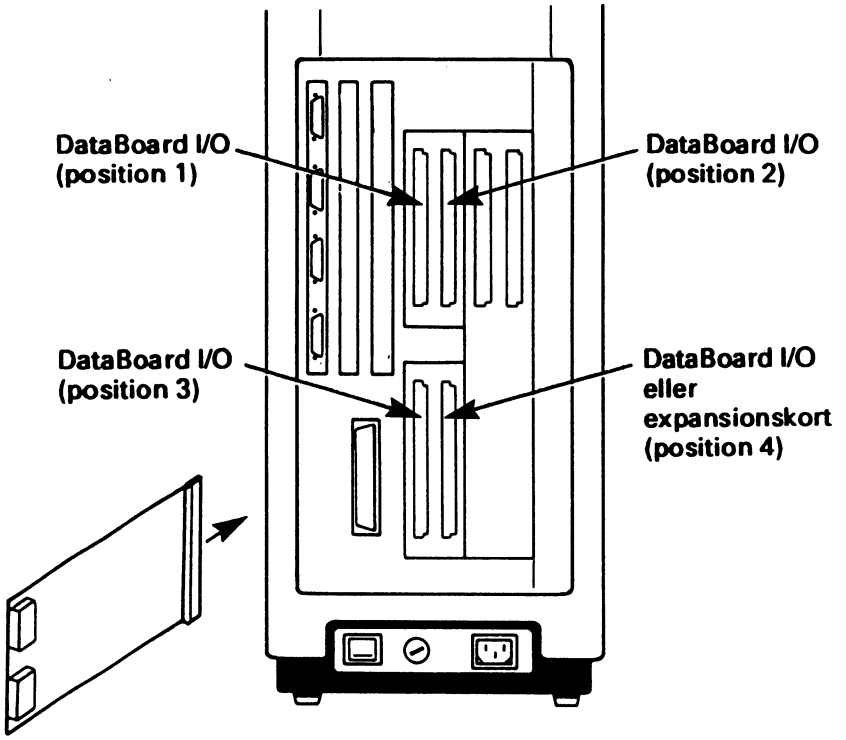
Antal kort: 4

Begränsningar:

Endast den nedre högra kontakten (markerad i figuren) ska användas för expansionskort (4203-00) till yttre rack.

Om expansionskort används i nedre högra positionen får inte övre vänstra positionen användas för kort som använder avbrottssignalen.

Om det tvåkanaliga DataBoardkortet 4118 används i nedre högra positionen får inte nedre vänstra positionen användas för kort som använder avbrottssignalen.



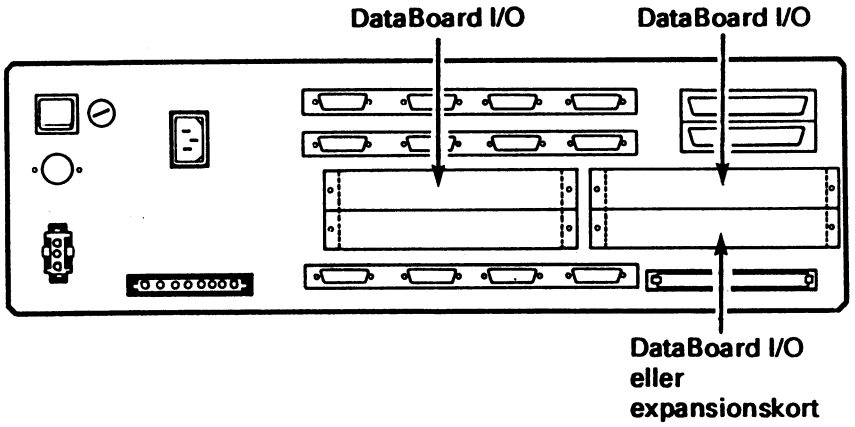
DataBoard-positioner i DS90-11:

Antal kort: 4

Begränsningar:

Endast den nedre högra kontakten (markerad i figuren) ska användas för expansionskort (4203-00) till yttre rack.

Om expansionskort eller det tvåkanaliga DataBoardkortet 4118 används i nedre högra positionen får inte övre vänstra positionen användas för kort som använder avbrottssignalen.



DataBoard-positioner i DS90-20:

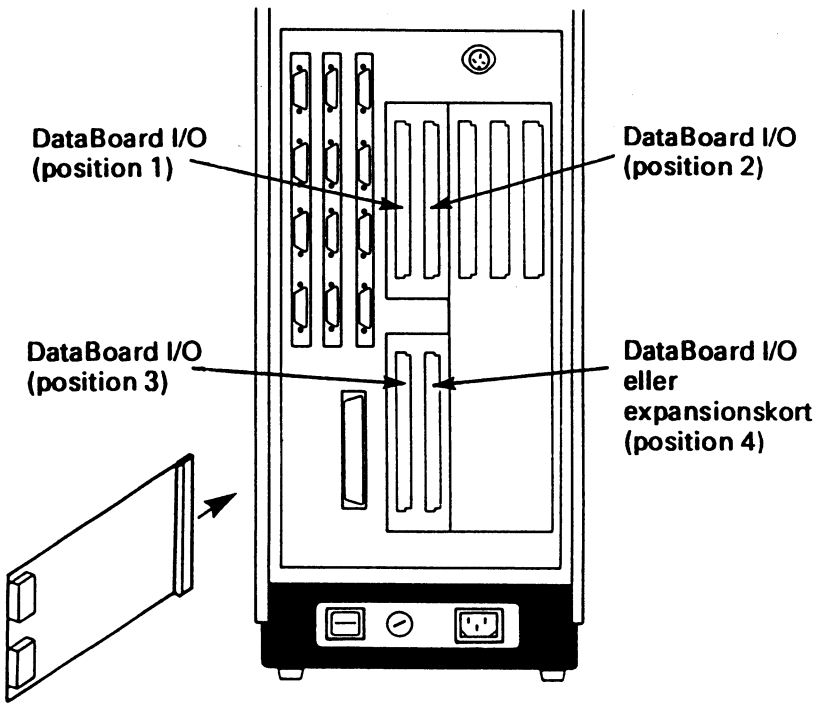
Antal kort: 4

Begränsningar:

Endast den nedre högra kontakten (markerad i figuren) ska användas för expansionskort (4203-00) till yttre rack.

Endast en kanal (kanal 0) kan användas på det tvåkanaliga DataBoard-kortet 4118.

Inga DataBoard-kort som kräver DMA kan användas.



DataBoard-positioner i DS90-21:

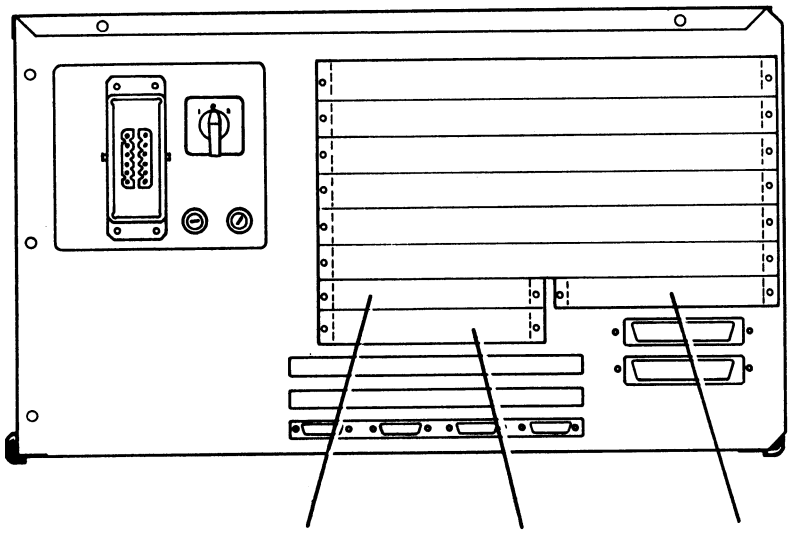
Antal kort: 3

Begränsningar:

Endast den övre högra kontakten (markerad i figuren) ska användas för expansionskort (4203-00) till yttre rack.

Endast en kanal (kanal 0) kan användas på det tvåkanaliga DataBoard-kortet 4118.

Inga DataBoard-kort som kräver DMA kan användas.



Anslutning av VME-kort

Beroende på vilken modell det gäller kan upp till sex VME-kort anslutas till expansionskontakterna i systemet. VME-korten ska ha dubbel Europakortsstorlek och en kontakt för 16 data- och 24 adressledningar.

Detta gäller inte DS90-11 och DS90-10E, som saknar VME-expansionsrack.

Anslutningen av VME-kort sker med spänningen frånslagen och korten sättes in med komponentsidan åt höger (I DS90-21 uppåt).

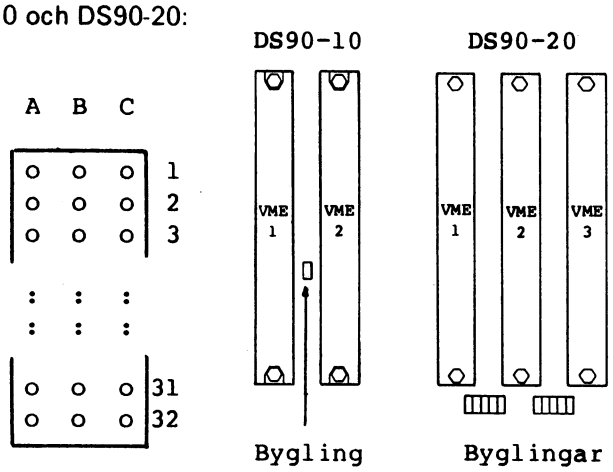
Om mindre än två VME-kort används ska de sättas i de vänstra (eller nedersta) positionerna, första kortet i VME-1, andra i VME-2 etc., om alla byglingar på bakplanet är öppna, vilket de normalt ska vara.

* Normalt är alla byglingar öppna om inte annat sägs nedan.

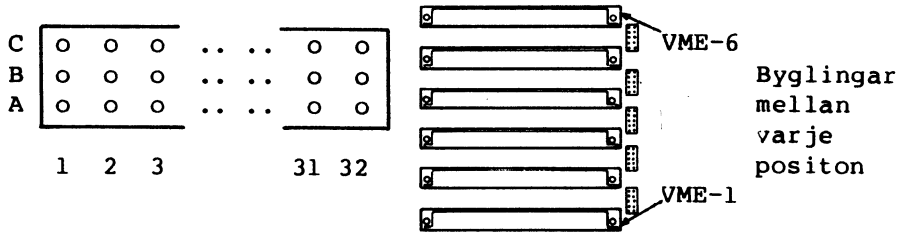
* Om någon VME-position är tom medan kort finns i en annan position med högre nummer, sluts byglingarna mellan den tomma positionen och nästa VME-position.

VME-busskontakten består av en tre-radiga Europakontakt, med stift som sitter enligt följande figurer. Antalet beror av datormodell.

I DS90-10 och DS90-20:



I DS90-21:



15.7.2 DataBoard generell I/O

Generell DataBoard I/O

/dev/DBinoutb är en drivrutin för direkt åtkomst till DataBoardbussen och de kort som är anslutna dit. Vid skrivning och läsning via **/dev/DBinoutb**, ges kortvalsadressen på DataBoardbussen och strobvärdet som en kombinerad enhetsadress.

Drivrutinen för /dev/DBinoutb har det primära enhetsnumret **3** (major device), och det sekundära enhetsnumret **8** (minor device).

Åtkomst till fysiska enheter kan normalt bara ske via operativsystemet (supervisor-mode), men i systemet finns möjlighet att definiera accesstillstånd i användarmod (usermode) så att utvalda DataBoardkort kan användas direkt av användarprogrammet.

Drivrutiner för avbrottshantering

/dev/DBIdrv är en drivrutin som kan användas för buffrad avbrottsstyrd utmatning av data till DataBoard-bussen. Vid access skrivs både intruktioner och data till rutinen.

/dev/DBIdrv har primärt enhetsnummer **13** (major device) och sekundärt enhetsnummer **0** (minor device).

Med utvecklingspaketet följer ett bibliotek med C-rutiner som kan användas vid egen programmering. Med dessa kan hanterare utvecklas med snabb access till DataBoardbussen.

För detaljer hänvisas till kapitlet om DataBoard i **PH**.

Denna drivrutin används bl a vid uppkoppling mot DataBoard-kort 4121 vilket är ett styrkort för Ethernet.

15.7.3 Generella drivrutiner för VME-bussen

/dev/VMEdrv är en drivrutiner som kan användas för buffrad avbrottstyrd access till VME-bussen. Rutinen kan användas för att sända kommandon till slavenheter på VME-kort och ta emot avbrottssignaler och data från VME-bussen.

Drivrutinen för /dev/VMEdrv har det primära enhetsnumret **12** (major device) och det sekundära enhetsnumret **0** (minor device).

För detaljer hänvisas till **PH**.

15.7.4 SP1/Centronics gränssnitt

I operativsystemet finns drivrutiner förberedda för anslutning av skrivare enligt gränssnitten SP1 och Centronics. Men för att dessa ska kunna användas måste dessutom anpassningskorten för dessa gränssnitt finnas installerad i maskinvaran. För SP1 heter sändarkortet DataBoard 4015 och för mottagarkortet DataBoard 4016. Ska ett Centronicsgränssnitt användas behövs anpassningskortet DataBoard 4001.

Kortadressen för anpassningskorten måste ligga mellan 00 och 63 (decimalt). Och då korten är enkelriktade måste även drivrutinen vara enkelriktad. D v s den får endast användas för läsning eller skrivning.

Om två datorer med SP1 gränssnitt kopplas samman med varandra kan de användas för överföring av data mellan datorerna. Överföringen kan då ske helt utan förvanskning, s k raw-mode. Raw-mode sänder 8-bitars data utan att ta någon hänsyn till vilka data som skickas. Överföring kan också ske i icke raw-mode. Då sänds block av ASCII-tecken där varje block avslutas med ett ASCII LF.

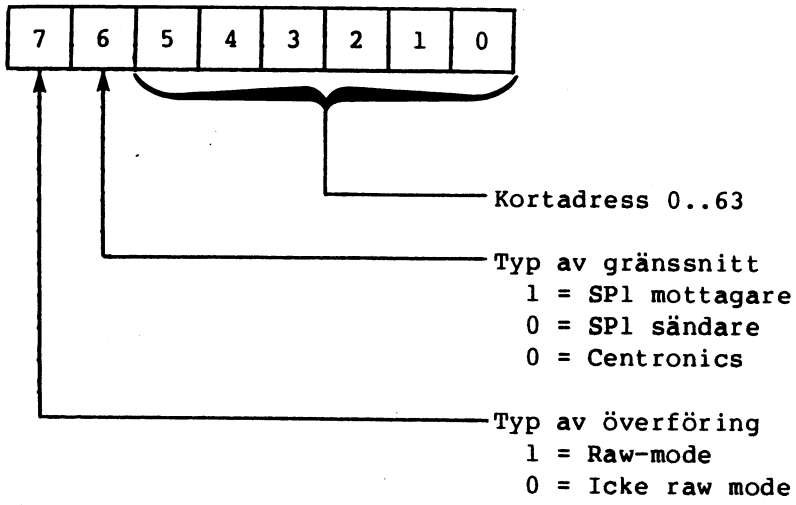
När drivrutinen används för att sända tecken till en skrivare bör icke raw-mode användas. Detta för att vissa specialtecken måste expanderas för att vara skrivbara på alla skrivare. T ex måste TAB expanderas till ett antal mellanslag för att många skrivare ska kunna hantera dem.

Icke raw-mode är speciellt lämpligt för överföring av ASCII-tecken mellan en D-NIX maskin och en icke D-NIX/Unix maskin. I D-NIX/Unix används bara ASCII NL för att markera slutet på en rad. I de flesta andra system används ACII CR/LF för att markera radslutet. I icke raw-mode sker konverteringen av NL till CR/LF automatiskt vid sändning och vid mottagning sker konverteringen i andra riktningen.

Generering av drivrutin

Drivrutinen **/dev/sp1** som finns i systemet har primärt enhetsnummer 7 (major device). Det sekundära enhetsnumret (minor device) är beroende av kortadress samt hur drivrutinen ska användas.

Det sekundära enhetsnumret byggs upp som ett 8-bits-värde, där bitarnas betydelse framgår av nedanstående figur:



Exempel:

```
Ett SP1 gränssnitt med  Kortadress         =  58
kortadress 58 för       Typ av gränssnitt  =   0
sändning i raw-mode.    Typ av överföring  = 128
```

Sekundärt enhetsnummer blir alltså 58 + 128, dvs **186.**

Exempel:

```
Ett SP1 gränssnitt med   Kortadress         =  63
kortadress 63 för mot-   Typ av gränssnitt  =  64
tagning i icke raw-mode. Typ av överföring  =   0
```

Sekundärt enhetsnummer blir alltså **127.**

Exempel:

```
Ett Centronics gräns-    Kortadress         =  10
snitt med kortadress 10  Typ av gränssnitt  =   0
för icke raw-mode.       Typ av överföring  =   0
```

Sekundärt enhetsnummer blir alltså **10.**

15.7.5 Extra fasta skivminnen

Systemet är försedd med en anslutningskontakt för att ansluta extra yttre skivminnen. Yttre skivminnen skall ha det primära enhetsnumret **5** (major device). Det sekundära enhetsnumret (minor device) beror av vilken datormodell som används och är olika om flera yttre skivminnen ansluts.

Till den yttre SASI/SCSI-porten kan skivminnen eller andra enheter, som kan styras via SASI/SCSI, anslutas. Antalet enheter som kan anslutas varierar mellan olika datormodeller och listas i början av kapitel 15. Till samma SASI/SCSI-port är normalt den interna kassettstreamern ansluten.

Det första yttre skivminnet bör ha det logiska enhetsnamnet **/dev/si2**.

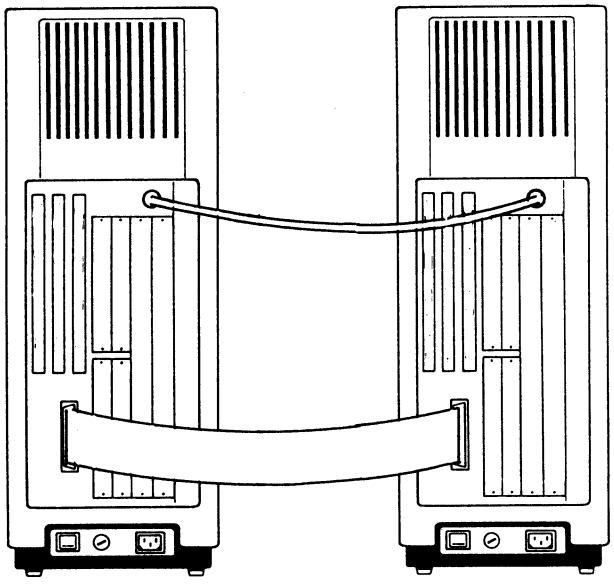
Vid systemstart från den första yttre winchesterenheten via systemets bootstrap PROM anges kanalnummer genom enhetsnamnet **si(X,0)**, där X är olika för olika system enligt nedan.

System	Sekundärt enhetsnummer	Enhetsnamn i laddnings-programmet	Enhetsnamn i D-NIX
DS90-11	18	si(18,0)	/dev/si2
DS90-10	2	si(2,0)	/dev/si2
DS90-10E	2	si(2,0)	/dev/si2
DS90-20	2	si(2,0)	/dev/si2
DS90-21	2	si(2,0)	/dev/si2

Yttre skivminnen ansluts mekaniskt till de olika datormodellerna enligt följande figurer.

Expansionsenhet med yttre skivminne

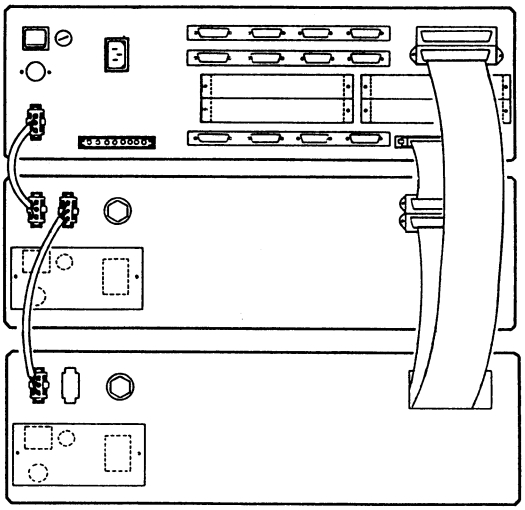
DS90-10/-10E/-20



DS90-11
Datorenhet

Massminnesenhet

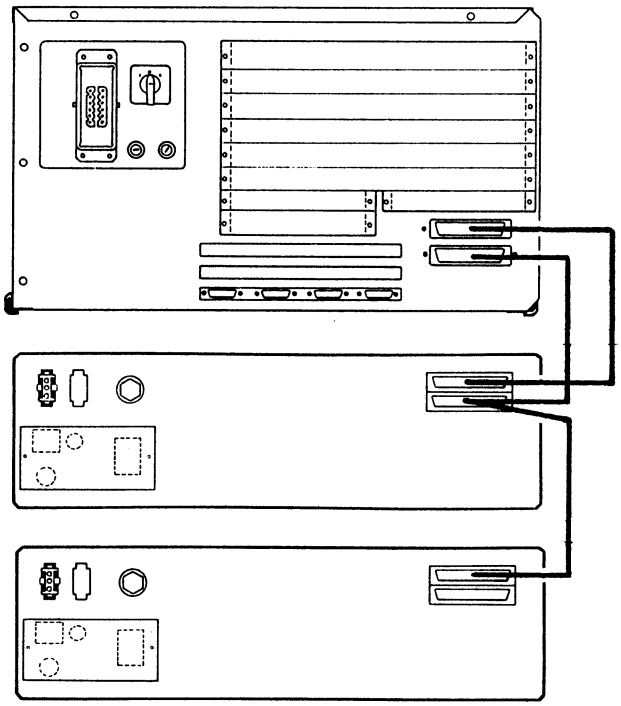
Expansionsenhet
med första yttre
skivminne



DS90-21
Datorenhet

Massminnesenhet

Expansionsenhet
med första yttre
skivminne



15.7.6 Extra diskettenheter

För anslutning av extra diskettenheter till systemet används Data-Boardkortet 4112 och tillhörande drivprogram.

15.7.7 Extra V.24 serieanslutningar

Systemet kan förses med upp till tolv seriella V.24 (RS232S) kanaler om två terminalexpansionkort (5172-10) monteras i datorn tillsammans med CPU-kortet. Varje kortenhet som ansluts till datorn är försedd med fyra V.24 anslutningar.

Ytterligare expansion sker genom DataBoard eller VME-kort.

Dessa anslutningar kan användas för terminaler och skrivare eller för anslutning av kommunikationslinjer med asynkrona protokoll.

CPU-kortet har fyra portar:

Enhetsnamn	Primärt enhetsnummer	Sekundärt enhetsnummer
/dev/console	1	1
/dev/lp	1	0
/dev/tty02	1	2
/dev/tty03	1	3

Det första extra kortet har:

Enhetsnamn	Primärt enhetsnummer	Sekundärt enhetsnummer
/dev/tty04	1	4
/dev/tty05	1	5
/dev/tty06	1	6
/dev/tty07	1	7

Det andra extra kortet har:

Enhetsnamn	Primärt enhetsnummer	Sekundärt enhetsnummer
/dev/tty08	1	8
/dev/tty09	1	9
/dev/tty10	1	10
/dev/tty11	1	11

15.7.8 Automodem 5124, 1200/75 eller 300 Baud

5124 är ett modem i form av ett DataBoardkort för överföringshastigheten 300 Baud dubbelriktat, eller 1200/75 eller 75/1200 baud split speed. Det kan anslutas till systemet och ger då möjlighet att ringa in till systemet som då automatiskt svarar. För att detta ska fungera krävs att kortet ansluts med primärt enhetsnummer **8** (major device) och sekundärt enhetsnummer **128+64+kanalval** (minor device). Kanalval är den kortadress som byglats på 5124 enligt medföljande beskrivning. Endast kanalval 0..7 är tillåtet. Överföringshastigheten och andra parametrar byglas på 5124-kortet.

För att ringa ut med hjälp av modemkortet 5124 krävs manuell nummerslagning och att andra byglingar görs på kortet.

Exempel:

Med kanalval = 2 blir sekundära enhetsnumret (minor) = **194**. För att skapa enheten ges kommandot:

```
mknod /dev/modem c 8 194
```

Nedan följer två byglingsexempel, för 75/1200 baud för inloggning via modemet samt för 1200/75 baud för att ringa ut till en annan datorn med hjälp av kommandot **cu**.

Exempel 1: Inloggning via modemet.

För att få en terminal att fungera mot 5124 lägges följande in i /etc/inittab och byglingar på 5124 sker enligt nedan.

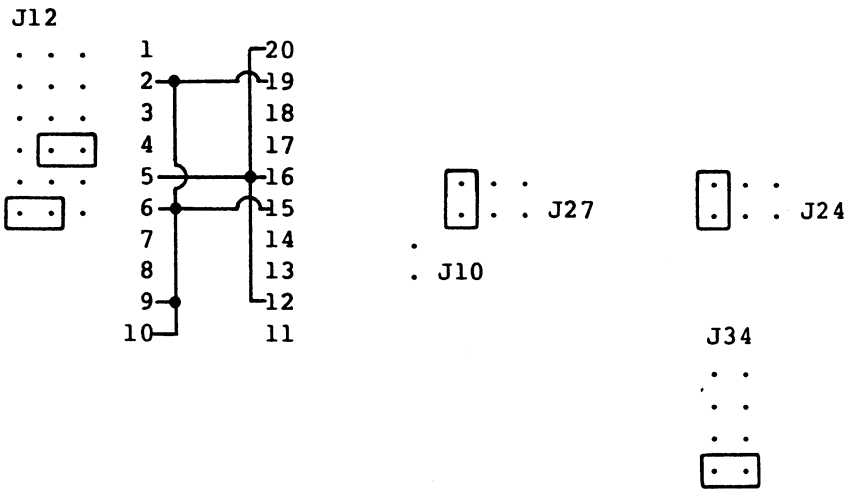
```
m2:2:respawn:nice -16 /etc/getty /dev/modem 1200
```

Pekaren 1200 ovan anger att parametrar enligt raden med 1200 i filen /etcgettydefs skall användas, men de parametrar som byglas fast på kortet 5124 ignoreras av drivprogrammet.

Kortet ska vara Byglat enligt nedan.

Auto-answer, 8 databitar utan paritet, och 1 stoppbit. Split-speed 1200 Baud sändning, 75 Baud mottagning.

Med denna konfiguration kan man ringa in till datorn och logga in som användare.



Se kortbeskrivning 5124 för ytterligare information om byglingar och andra maskinvaruberoende funktioner.

Exempel 2: För att ringa ut med **cu**-kommandot via modemet.

För att kunna ringa ut via **cu**-kommandot och 5124, skall inte modemenheten aktiveras i /etc/inittab, nummerslagningen skall ske manuellt och **cu**-kommandot skall ej ges förrän svarston erhålls från den uppringda datorn. 5124-kortet skall byglas enligt nedan.

Uppringande (manuellt), 8 databitar utan paritet, och 1 stoppbit. Split-speed 75 Baud sändning, 1200 Baud mottagning.

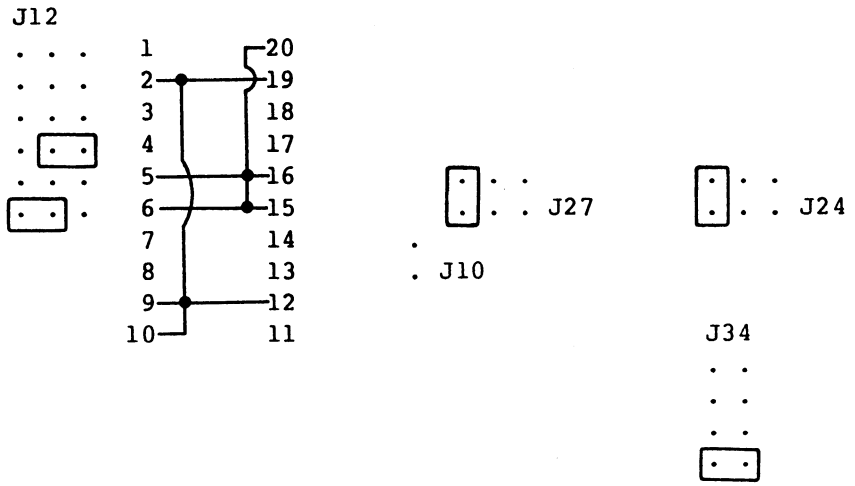
Ett modem uppsatt enligt denna konfiguration kan kommunicera med ett annat som satts upp enligt exempel 1 ovan.

För att ringa upp görs följande:

* Skriv in **cu**-kommandot, men vänta med att trycka på <RETURN> för att starta kommandot. Hastighetsparametern (-s1200) ignoreras av 5124-drivprogrammet och behövs egentligen inte.

```
cu -l/dev/modem -s1200 dir        (Ge ännu inte <RETURN>)
```

* Ring upp manuellt och vänta på svarston från den uppringda datorn. Då svarston kommer, trycks <RETURN>-tangenten ner så att **cu**-kommandot startar och tar linjen. Därefter sker kommunikationen direkt från tangentbordet. Med <RETURN> ü . <RETURN> avslutas **cu**-kommandot, varvid rutinen släpper linjen (lägger på luren).



Se kortbeskrivning 5124 för ytterligare information om byglingar och andra maskinvaruberoende funktioner.

15.7.9 TTY-portar, 4117 och 4118

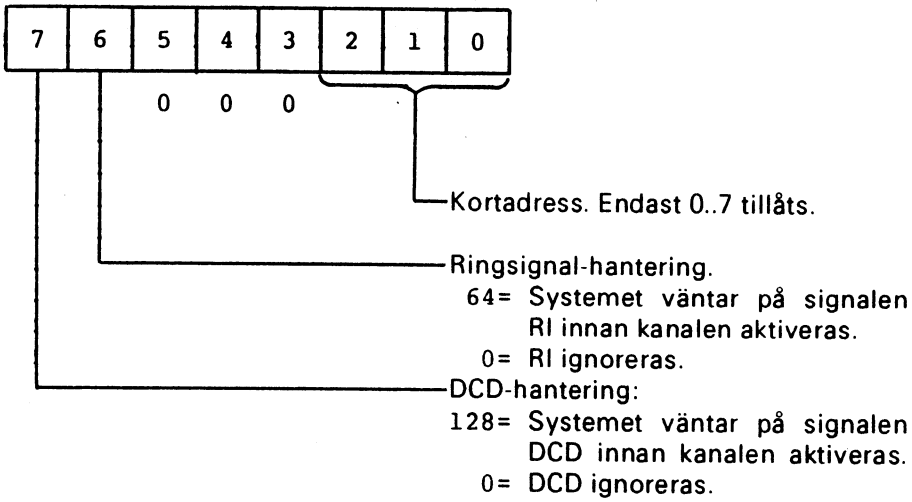
4117 respektive 4118 är två kort som ger extra terminalportar. Korten ska vara anslutna med enhetsnamn och nummer enligt nedan.

OBS! Med 4118 gäller vissa restriktioner.

/dev/uann

nn = Kortadress på kortet (Card select).

Det primära enhetsnumret ska vara 8 (major device) och det sekundära enhetsnumret (minor device) är uppbyggt som ett 8-bits-värde, där bitarnas betydelse framgår av nedanstående figur:



Obs! Endast kortval 0..7 ('nn' i enhetsnamnet) kan användas. Kortvalet byglas på kortet enligt medföljande kortbeskrivning.

Då både bit 7 och bit 6 är satta (minor=128+64+kanalval) kommer drivprogrammet först att vänta på RING signal och sedan vänta på DCD. Detta är mycket användbart vid modem hantering då ett externt modem används för att kunna ringa in till systmemet.

Exempel: Ett 4117-kort med kortadress 1 heter /dev/ua01. Utan ringsignal eller DCD-detektering används då sekundära enhetsnumret 1. För att skapa enheten ges kommandot:

```
/etc/mknod /dev/ua01 c 8 1
```

För att kunna logga in från en terminal ansluten till kortet lägges följande rad in i filen /etc/inittab. Pekaren (9600) anger vilken tabell i filen /etc/gettydefs som ska användas, men de parametrar som byglas fast på kortet 4117/4118 ignoreras av drivprogrammet.

```
ul:2:respawn:nice -16 /etc/getty /dev/ua01 9600
```

15.7.10 Kommunikation

Kommunikation med synkrona protokoll till andra datorer och även lokala (LAN) eller distribuerade nätverk (WAN) utnyttjar DataBoardbussen som finns i systemet.

Ett flertal kort i DataBoardserien understöds direkt av operativsystemet. De drivrutiner som finns kan aktiveras enligt tidigare beskrivning.

Systemen är i ursprungsskick försedd med kortplatser för DataBoard, och en yttre expansionsrack kan anslutas vid behov.

OBS! Vid installation och in- respektive urkoppling av kort och kablage anslutna till DataBoard bussen måste spänningen till systemet kopplas ifrån både i datorn och i anslutna yttre enheter.

5.7.11 Bandstation med 1/2 tums band - /dev/mt0

För bandstationer skall enhetsnamnet vara **/dev/mt0**.

Primära enhetsnumret (major device) ska vara **6**, medan det sekundära (minor device) normalt är **0**.

Yttre bandstationer kan anslutas på olika sätt.

I DS90-11 kan DataBoard-kortet 4104 användas för direktanslutning till bandstationer med Pertec styrenhet.

I DS90-10/-20/-21 kan Ett VME-kort, med styrenhet för bandstationer, användas för direktanslutning till bandstationer med Pertec styrenhet.

Utvärdering av dokumentation:

Handbok Systemadministration 089-9821-10 (A)

Vi är mycket intresserade av vad Du tycker om dokumentationen och hur Du tycker att den skulle kunna förbättras. Vi vore därför mycket tacksamma om Du använde några minuter av Din tid för att besvara frågorna nedan och returnera detta papper till oss.

1. Är det lätt att hitta den information Du söker? Är det någon information Du behöver som Du inte kan hitta?

_____

2. Är texten lättläst och lätt att förstå? Ange gärna sida och stycke om Du har något exempel på svårförståeligt avsnitt.

_____

3. Är uppbyggnaden av dokumentationen logisk? Bör några avsnitt byta plats, flyttas eller liknande?

_____

4. Finns det tillräckligt många exempel och beskriver de rätt delar av informationen?

_____

5. Hur tycker Du att dokumentationen skulle kunna förbättras, ge gärna exempel?

_____

_____

_____

_____

_____

Namn:_____ Företag: _____
(Frivilliga uppgifter)

På baksidan av detta formulär finns vår adress förtryckt. Vik formuläret på mitten och häfta eller tejpa ihop det så kan det skickas som det är med posten.

Tack för hjälpen!

DataIndustrier DIAB AB
Dokumentation
Box 2029
183 02 TÄBY