D-NIX 5.3
Referenshandbok

Version B.1 91-03-20

089-9512-10

Inledning

Intro(1)

Referensblad

Diab Data AB
Box 2029
183 02 TÄBY
☎ 08-638 94 00

DIAB DATA

INLEDNING

1.1 D-NIX operativsystem

I denna handbok beskrivs kommandon till operativsystemet D-NIX, som är implementerat på Diabs datorer. Operativsystemet D-NIX är helt kompatibelt med AT&T Unix System V Version 3, och innehåller samma funktioner. Men en del kommandon har lagts till och andra kommandon har fått fler tillval.

Handboken gör inga anspråk på att ge grunderna till Unix eller D-NIX. Det finns många bra böcker om Unix system. Vi rekommenderar UNIX SYSTEM V Primer av Waite, Martin och Prata.

Det bästa sättet att lära sig om Unix är genom övning på terminalen. Var inte rädd för att göra misstag, det är av dem du lär dig.

1.2 Kommandoformat

Kommandona i Unix-sfären är något speciella eftersom de med få undantag är kortfattade och inte talar om för användaren vad de utför, tex ls, su, nice. I regel är kommandona ett sammandrag av ett eller flera ord, t ex ls - list contents of directory, cat - concatenate (sammanfoga).

Kommandona skrivs med små bokstäver. Ett kommando som skrivs med versaler kan inte utföras om inte kommandonamnet också är definierat med versaler i systemet.

Alla kommandon givna i D-NIX tolkas av kommando-processorn Shell. Shell är en förbindelse mellan användaren och D-NIX operativsystem. Med Shell kan man omdirigera in- och utdata, tilldela variabler och göra substitutioner (utbyten) på en kommandorad innan ett kommando utförs. I interaktivt mod används Shell som programmeringsspråk, vilket gör det möjligt för användaren att skapa egna kommandon genom att kombinera flera kommandon i kommandofiler.

Standardformatet på ett kommando är en sekvens av ord, separerade med ett eller flera mellanslag. Alla kommandon utförs efter nedtryckning av RETUR. Flera kommandon kan anges på samma rad om de separeras med ett semikolon ';'

Det första ordet är kommandot och resterande ord är argument till kommandot. Argumenten får ej innehålla mellanslag, men man kan ha flera mellanslag både före, efter och även mellan argumenten. Om ett argument innehåller mellanslag måste det omges av " " eller ' '.

I regel består argumenten av ett av följande uttryck:

Tillval Innehåller en eller flera bokstäver - vanligen föregångna av minus-tecken (exempel -al). Antingen modifierar tillvalet kommandot eller anger exakt vilken funktion kommandot skall utföra.

Uttryck Ett uttryck ('expression') består av en teckensträng som kommandot kan använda.

Filnamn/Bibliotek

Ger namnet på den fil som kommandot ska behandla på ett eller annat sätt. Filnamnet kan bestå av 14 eller färre tecken.

Den vanligaste ordningsföljden för ett kommando är:

kommando tillval uttryck filnamn/bibliotek

För den exakta ordningsföljden hänvisas till kommandot ifråga.

NAMN Anger namnet på kommandot och en mycket kortfattad beskrivning.

SYNTAX Visar hur kommandot används. Här anges också möjliga tillval och argument.

FUNKTION Ingående beskrivning av kommandot och hur det fungerar.

TILLVAL Beskrivning över vilka tillval som existerar och deras funktioner.

FILER Namn på de standardfiler som används av eller kan påverkas av kommandot.

EXEMPEL Exempel på användningen av kommandot.

HÄNVISNING Refererar till snarlika kommandon. Referenser kan även vara till andra handböcker till exempel handböckerna för utbyggnadspaketet.

FELMEDDELANDEN

Beskriver i vilka fall som felmeddelande skrivs ut.

ANMÄRKNING

Anger om det är något särskilt att tänka på.

Under syntax används följande konventioner:

- Hakparenteser [] runt ett argument, indikerar att det inte är obligatoriskt att ange detta argument.
- Ordet "namn" anger att här kan både ett filnamn eller ett bibliotek anges.
- När ett argument kan upprepas hur många gånger som helst betecknas detta med "...".

1.3 Speciella uttryck

Det finns en del specialuttryck som är unika för D-NIX. De vanligaste beskrivs nedan:

Super-User En privilegierad användare som har obegränsad åtkomsträtt till alla delar av systemet.

Standard Output

D-NIX skickar resultatet av ett kommando till en fil som heter standard output. Normalt är denna riktad mot terminalen.

Standard Input

Standard Input är den plats varifrån programmet förväntar sig att läsa data. Normalt är detta från tangentbordet.

Standard Error

Felmeddelanden från kommandon skickas normalt till standard error. Normalt är denna fil riktad mot terminalen.

Specialfiler Detta är specialfiler som står i förbindelse med de externa enheterna såsom terminaler, radskrivare, massminnen. Filerna läses och skrives på liknande sätt som vanliga filer med hänsyn tagen till de yttre enheternas restriktioner.

Wildcards, Metatecken eller Jokertecken

När filnamn ges på en kommandorad har vissa tecken, wildcards, speciella betydelser. Wildcards kan även heta metatecken eller jokertecken. Shell kommer att avkoda dessa innan argumenten ges till kommandot. Ett argument med metatecken ersätts av flera argument, ett för varje filnamn i filsystemet som matchar tecknen. Tecknen *, ? och [.....] är generella metatecken. Om dessa skall användas utan sin speciella betydelse måste de antingen föregås av tecknet \ eller sättas inom apostrofer '.....'. Se kommandot sh för detaljer.

Pathname eller Sökväg

Ett pathname eller en sökväg är ett fullständigt fil- eller biblioteksnamn, vilket innehåller hela sökvägen genom alla bibliotek från rootbiblioteket.

Mounta ett filsystem

Ett eller flera filsystem på yttre skivminnen eller inom en fil kan anslutas (kopplas), till root-filsystemet, så att det yttre filsystemet behandlas som om det vore ett bibliotek inom root-filsystemet. Denna anslutning görs med kommandot mount, varvid filsystemet säges vara mountat.

Utgångsstatus

När ett kommando avslutas returnerar det alltid ett värde till systemet, utgångsstatus. Detta värde är normalt noll (0). Andra värden kan returneras och visar oftast att ett fel har upptäckts, men ibland anger utgångsstatus något resultat från kommandot.

Boota ett system

När en dator startas, måste operativsystemet laddas från ett skivminne eller en diskett och därefter startas. Denna procedur kallas att boota systemet.

INLEDNING

NAMN

intro - introduktion till kommandon och applikationsprogram

FUNKTION

I denna handbok beskrivs D-NIX kommandon i alfabetisk ordning. Nedan följer regler och annat som kan vara bra att läsa innan man börjar slå i den här referenshandboken.

REFERENSHANDBOKENS KOMMANDOSYNTAX

Om inte annat beskrivs skall kommandon köras och ges tillval och argument som det är beskrivs nedan.

kommando [-tillval] [argument]

[ ]	Omger ett tillval eller ett argument som inte är nödvändigt.
...	Anger att optionen eller argumentet kan förekomma mer än en gång.
kommando	Namnet på en exekverbar fil.
tillval	(Föregås alltid av ett "-".) Består antingen av en tillvalsbokstav eller en tillvalsargumentbokstav + ett tillvalsargument.
tillvalsbokstav	En ensam bokstav som representerar ett tillval som inte har ett argument. Observera att mer än en tillvalsbokstav kan följa efter ett "-" (Regel 5, nedan).
tillvalsargumentbokstav	En ensam bokstav som representerar ett tillval som behöver att tillvalsargument.
tillvalsargument	Ett tillvalsargument (teckensträng) som hänger ihop med en tidigare tillvalsargumentbokstav. Observera att när det blir fler än ett tillvalsargument måste dessa spareras med "," eller mellanslag/tab och omges av citationstecken. (Regel 8, nedan.)
argument	Sökväg (eller annat argument) som inte börjar med ett "-", eller ett "-" som indikerar standard input.

KOMMANDOSYNTAX STANDARD (REGLER)

Dessa kommandosyntaxregler följs inte av alla nuvarande kommandon men alla nya kommandon skall följa dem. *getopts(1)* bör användas av alla shell-procedurer för att avkoda positionsparametrar och kontrollera tillåtna tillval. *getopts(1)* stödjer reglerna 3-10 nedan. Kommandona måste själva se till att övriga regler efterlevs.

1. Kommandonamn måste vara melllan två och nio tecken långa.
2. Kommandonamn får endast innehålla gemena bokstäver samt siffror.
3. Tillval får endast vara ett tecken långa.

4. Tillval måste föregås av ett minustecken "-".
5. Tillval utan argument får skrivas tillsammans efter ett ensamt "-".
6. Det första tillvalsargumentet som följer efter ett tillval måste föregås av ett mellanslag eller en tab.
7. Tillvalsargument kan ej uteslutas.
8. Om flera tillvalsargumnet följer på ett tillval måste dessa antingen separeras med komma "," eller med tab eller mellanslag och placeras inom citationstecken. Exempel: -o xxx,z,yy eller -o "xx z yy".
9. Inga tillval får komma efter argument på kommandoraden.
10. "--" markerar slut på tillval. Detta kan användas om något argument börjar med "-"
11. Tillvalens inbördes ordning skall inte ha någon betydelse.
12. Argumentens inbördes ordning kan påverka deras betydelse, detta bestäms av kommandot ifråga.
13. "-" föregånget och följt av tab/mellanslag, skall bara användas för betydelsen standard input.

TECKENUPPSÄTTNINGAR

Vissa tecken som finns i denna handbok finns ej på en del tangentbord, dessa skall då ersättas med följande:

}	=>	å
]	=>	Å
{	=>	ä
[	=>	Ä
|	=>	ö
\	=>	Ö
~	=>	ü
^	=>	Ü
‘	=>	é
@	=>	É
$	=>	¤
#	=>	£ eller §

HÄNVISNING

getopts(1).

exit(2), wait(2), getopt(3C) i *Programmer's Reference Manual*.

FELMEDDELANDEN

Varje kommando returnerar två statusbytes, där den ena ges av systemet och anger varför kommandot avslutades, och den andra (vid "normalt" av-

slut) ges av kommandot själv (jämför *wait(2)* och *exit(2)*). Den första byten är 0 vid normalt avslut, den andra är vanligen 0 vid normalt avslut, och skild från 0 för att indikera problem, såsom felaktiga parametrar eller data. Statusbyten beskrivs enbart där funktionen skiljer sig från den normala.

ANMÄRKNING

Vissa kommandon producerar oväntade resultat när de kör filer som innehåller NULL-tecken. Dessa kommandon behandlar ofta inskriven text som textsträngar och uppfattar då ett NULL-tecken som strängslut.

NAMN

accept, reject - tillåter/hindrar utskrift med lp till angivna skrivare.

SYNTAX

/usr/lib/accept [-V] destinationer

/usr/lib/reject [-V] [-r[*orsak*]] destinationer

FUNKTION

Accept tillåter *lp* att ta emot utskrifter riktade mot destinationer. Destination kan antingen vara en skrivare eller en klass (grupp) skrivare. Använd *lpstat* för att se status för destinationerna.

Reject hindrar *lp* från att ta emot utskrifter riktade till destinationer. En destination kan antingen vara en skrivare eller en klass (grupp) av skrivare. Använd *lpstat* för att se status för destinationerna.

Följande tillval kan användas med *reject*. **-V** används även med *accept*.

TILLVAL

-V Skriver ut versionsnumret för kommandot.

-r*orsak* Används med *reject*. Anger orsak varför *lp* inte accepterar utskrifter. Denna orsak gäller för alla skrivare fram till nästa **-r** tillval. Orsaken rapporteras av *lp* när användare sänder utskrifter till nämnda destinationer samt av *lpstat*. Om tillvalet **-r** inte finns eller om **-r** tillvalet anges utan orsak används en standardtext som orsak.

FILER

/usr/spool/lp/*

HÄNVISNING

enable(1), lp(1), lpadmin(1M), lpsched(1M), lpstat(1), reject(1M)

ANMÄRKNING

Om *orsak* består av mer än ett ord, separerade av mellanslag, måste orsak omges av apostrofer. Exempel:

```
reject -r'Detta är orsaken' main
```

NAMN

badblk - Ersätter dåliga sektorer

SYNTAX

/etc/badblk [-V] [-f] [-p] filsystem sektor sektor...

FUNKTION

badblk är ett kommando som används för att reparera ett filsystem på ett massminne, i vilket en sektor fysiskt skadats. Denna skall ersättas med en sektor utan fel. En sektor är en fysisk adress i filsystemet. Sektorstorleken är alltid 512 bytes, dvs adressen till en sektor är alltid *n**512, där *n* är ett sektornummer. Om blockstorleken på enheten är större än 512 bytes, tex på SMD-skivminnen, måste man räkna om den fysiska adressen till motsvarande 512 bytes sektornummer och ge detta som parameter till *badblk*.. En eller flera sektorer kan anges på kommandoraden.

badblk placerar numren på alla dåliga sektorer i en fil benämnd **/..systemfiles/badspots**. Samtidigt får användaren en felfri sektor i ersättning. Normalt kopieras det gamla innehållet över till den nya sektorn där så är möjligt.

Filsystem är ett enhetsnamn eller ett filnamn skapat med */etc/mkfs*. Filsystemet får inte vara "mountat" och måste vara rent. För att reparera det interna skivminnet måste systemet stängas av och startas manuellt på enanvändarnivå med BOOT-disketten som rootenhet. För en beskrivning se **Systemadministration**.

Kommandot */etc/fsck* skall utföras både före och efter kommandot *badblk*. Kommandot före rensar systemet och kommandot efter uppdaterar systemet och gör alla nödvändiga kompletteringar om badblk inte har kunnat kopiera innehållet av några systemsektorer.

TILLVAL

Följande tillval kan föregå filsystemets namn.

-V	Skriver ut versionsnumret på *badblk*.
-f	Tvingar fram ersättning av dåliga sektorer och rensar de nya sektorerna. Detta kan medföra problem om den ersatta sektorn är en del av en inodlista eller en indexsektor i en fil. Utför alltid kommandot *fsck* när tillvalet **-f** använts.
-p	Skyddar filsystemet mot skrivning.

FILER

/..systemfiles/badspots

EXEMPEL

Exempel 1:

Ett korrigerbart fel har hittats på den interna winchesterenheten **/dev/si0**, sektor 3210 med byteadressen 512*3210. Sektorn är inte tillförlitlig och måste ersättas. Systemet måste tas ned och startas igen manuellt med BOOT-disketten som rootenhet på enanvändarnivå (bootnivå 2).

```
/etc/fsck /dev/si0          Kontrollera enheten.
/etc/badblk /dev/si0 3210   Reparera.
/etc/fsck /dev/si0          Kontrollera en gång till.
```

Exempel 2:

En sektor med ett "hårt" fel har hittats på den interna winchesterenheten **/dev/si0**, sektor 1234. För att ersätta denna gör som exempel 1 men använd tillvalet **-f** i badblk-kommandot.

```
/etc/fsck /dev/si0          Kontrollera enheten.
/etc/badblk -f /dev/si0 1234    Reparera.
/etc/fsck /dev/si0              Kontrollera en gång till.
```

Glöm inte att köra *fsck* efter tillvalet **-f**, eftersom den nya sektorn är tom.

Exempel 3:

Ett dåligt filsystem på en diskett kan repareras direkt från den normala fleranvändarnivån. Disketten får inte vara "mountad". Anta att sektor 350 på en 5 1/4 tums diskett ska ersättas:

```
/etc/fsck /dev/mf0          Kontrollera enheten.
/etc/badblk -f /dev/mf0 350     Reparera.
/etc/fsck /dev/mf0              Kontrollera en gång till.
```

HÄNVISNING

/etc/fsck(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

Kommandot *badblk* kan inte användas i laddningsprogrammet (standalone) och är systemberoende.

NAMN

banner - skapar huvudtitlar, försättstitlar (posters)

SYNTAX

banner [-V] [-c *tkn*] [-e] [-E] [-f *fil*] strängar

FUNKTION

banner skriver ut argumenten (varje högst 10 tecken långt) med stora bokstäver på standard output.

Tecknen formas som standard enligt US-ASCII teckenfont (intern i *banner*-kommandot), men andra teckenfonter valda ur filer kan väljas. I en fil med en teckenfont används följande nyckelord:

Heigth n	Definierar höjden av 'banner'-tecknen
Width n	Definierar bredden av 'banner'-tecknen
Define c	Följs av ett antal rader som definierar utseendet av tecknet c. Antal rader får inte vara mindre än 'Height'.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *banner*
-c *tkn*	Formar tecknen med tecknet *tkn*. Som standard används **#**.
-e	Formar tecknen med tecknet självt.
-E	Formar tecknen med versalen av tecknet självt.
-f *fil*	Använd fonten definierad i filen *fil*. Utan **-f** används en intern fontdefinition enligt US-ASCII.

FILER

/usr/etc/banner.ascii	Font med ASCII-tecken
/usr/etc/banner.swe	Svensk teckenfont

HÄNVISNING

echo(1)

FELMEDDELANDEN

Inga

BANNER(1)

NAMN

basename - tar bort biblioteksnamn och/eller suffix ur pathname.

SYNTAX

basename [-V] sträng [suffix]

FUNKTION

basename tar i en sträng bort alla prefix som slutar med ett / och suffix, (om det finns några) och skriver resultatet på standard output. Resultatet blir filens "basnamn", dvs filens namn utan föregående bibliotek och utan suffix. Satt mellan ‘ ‘ används det i shellprocedurer för att skapa nya filnamn.

Det finns ett liknande kommando *dirname*, som tar bort det sista elementet i strängen och skriver den kvarvarande delen på standard output.

TILLVAL

-V Skriver ut versionsnumret av kommandot *basename*.

FILER

Inga

EXEMPEL

Exempel 1:

```
basename /usr/tf/brev.brv .brv
brev
```

Resultatet blir 'brev' eftersom trunkering av suffixet .brv begärdes vid kommandoanropet.

Exempel 2:

```
basename /usr/tf/brev.brv
brev.brv
```

HÄNVISNING

dirname(1), sh(1)

FELMEDDELANDEN

Inga

BASENAME(1)

NAMN

bootpar - ändrar bootparametrar i NVRAM

SYNTAX

/etc/bootpar [-V]

/sas/bootpar [-V]

FUNKTION

/etc/bootpar eller */sas/bootpar* används för att ändra vissa bootparametrar lagrade i NVRAM i datorns hårdvara.

Versionen */etc/bootpar* ska användas under operativsystemet av systemadministratören (super-user).

Versionen */sas/bootpar* ska användas från laddningsprogrammet, efter start i manuell mod till bootnivå 1.

bootpar är interaktivt och frågar användaren vilka parametrar som ska ändras. De aktiva parametrarna visas inom parentes och ändras inte om man enbart trycker RETUR.

Följande frågor bör besvaras. Om endast RETUR anges på varje rad, visas följande på skärmen, (eventuellt med andra parametervärden som aktiva parametrar). Om nya värden inmatas vid någon av dessa frågor och man svarar **j** (ja) på frågan Skriv till NVRAM, lagras de nya värdena och används nästa gång systemet startas upp.

```
Console baudrate    (9600)          Skall alltid vara 9600.
Boot level          (3)             AUTO-start nivå.
Boot device         (5,16)          Systemberoende.
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.
CPU clock freq.     (16670000)      Systemberoende.

Satisfied ? y                 (n => Ändra igen).
Write to NVRAM ? n                  (y => spara nya värden i NVRAM).
```

Console baudrate

väljer hastighet för huvudterminalen vid systemstart och i laddningsprogrammet. 9600 Baud skall alltid användas!

Boot level anger den bootnivå dit systemet startas upp när nyckeln på framsidan står i läget AUTO. 3 motsvarar fleranvändarnivån.

Boot device är den enhet där systemet startas upp om inte operatören väljer enhet manuellt. Värdena är beroende av systemet och bör normalt motsvara den interna winchester enheten.

CPU clock frequecy

är klockfrekvensen angiven i Hz. Värdet är beroende av maskinvaran.

TILLVAL

-V Skriver ut versionsnumret av kommandot *bootpar*.

FILER

/sas
/etc/timezone
/etc/passwd

HÄNVISNING

date(1), stty(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Detta kommando är systemberoende.

Tidigare versioner av *bootpar* hade ytterligare två parametrar. Dessa används ej längre men bör i äldre system vara satta enligt nedan. Jämför kommandot *date*.

```
Time zone           (60)      Relativt GMT.
US Daylight zone    (N)       Används ej. Skall vara N.
```

NAMN

bup - säkerhetskopiering av valda filer (backup)

SYNTAX

bup [-V] -[mwdak] filnamn ....

bup [-V] -[mwdak] -r *findargument*

bup [-V] -L

bup [-V] -l [-v] [-u*volymnamn*] [filnamn ...]

bup [-V] -e -u*volymnamn*

bup [-V] -n[mwd]

bup [-V] -b[mwda] [-cMv] [-u*volymnamn*] [-tk *volymstorlek*] [-tb *blockfaktor*] enhet

bup [-V] -x [-vM] enhet filnamn ....

FUNKTION

Kommandot *bup*, som är ett hjälpprogram, utför periodiskt återkommande säkerhetskopiering av specificerade filer. *bup* använder *tar*-kommandot för att göra själva säkerhetskopieringen. Kommandot *cron* används 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 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/volumenames** och de kommer att växla automatiskt. Loggfilen eller valda delar av den kan tas fram och visas med kommandot *bup* **-l**.

bup-kommandot skall utföras i tre steg:

1. Först lagras standard volymnamn i filen **/usr/lib/bup/volumenames** med hjälp av en editor, t ex *dmacs*.
2. Sedan körs *bup* för att lagra namnen på alla filer (eller bibliotek), som ska säkerhetskopieras, i kommandofilen **/usr/lib/bup/cmd**.
3. Som tredje steg görs själva kopieringen med *bup*-kommandot och en av följande tillval; **-bd**, **-bw**, **-bm** eller **-ba**, valfritt regelbundet aktiverat genom att använda kommandot *crontab*.

För att specificera filresp. biblioteksnamnen körs *bup* med ett av följande tillval; **-m**, **-w**, **-d** eller **-a**. Tillvalet **-m** används för filer som kopieras mån-

atligen, **-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 in fil- eller biblioeksnamnet i *bup*:s kommandofil **/usr/lib/bup/cmd** för senare användning då själva säkerhetskopieringen äger rum.

Fil- eller biblioteksnamnet kan specificeras antingen som normala sökvägsnamn (pathname), wildcards eller som argument till ett *find*-kommando (*bup* **-r** tillvalet) för att välja filer. *bup* kommer alltid att lägga till nuvarande bibliotek till alla filnamn som inte är fullständiga sökvägsnamn (pathname). Om namnet på ett bibliotek anges tages hela biblioteket. Om wildcards eller find-argument ges kommer de inte att bli utvärderade 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 kan listas med det kommandot *bup* **-L**.

Filen **/usr/lib/bup/volumenames** 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 eller tab-tecken. 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/volumenames**:

```
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 kommando-tillval till tar och listan med filer lägges till loggfilen tillsammans med volymnamnet.

```
tar -cvfSF enhet fil
tar -cvfbkSF enhet blockstorlek volymstorlek fil
```

I loggfilen **/usr/lib/bup/log** är varje fil, som lagrats på en kopieringsvolym, listad i ett format liknande kommandot *ls* **-l** med det fullständiga sökvägsnamnet (pathname) tillsammans med volymnamn för backup och tidpunkten för kopieringen. Endast den senaste tidsangivelsen för kopiering av varje fil sparas i loggfilen. Om samma fil finns på olika medier, finns den listad en gång per media. Loggfilen kan listas med kommandot *bup* **-l fil**.

TILLVAL

Allmänna tillval:

-V Skriver *bup*-kommandots versionsnummer och en kortfattad upplysning om användningen.

-v (Verbose=mångordig). I kombination med tillvalet **-b** eller **-x** skrivs normala listorna från tar ut på standard output. Kombinerad med tillvalet **-l**, skrivs även datum och tid när filen säkerhetskopierades.

-u *volymnamn* Specificerar volymnamn, som vanligtvis är ett bandnummer eller liknande.

När säkerhetskopiering görs lagras detta namn i loggfilen, och vi rekommenderar därför att bandet/disketten märks på samma sätt. Om tillvalet **-u** inte är specificerat och något av tillvalen **-bm**, **-bw** eller **-bd** används kommer *bup* att försöka hämta det volymnamn som anges i filen **/usr/lib/bup/volumenames**.

När filer återläses från kopian med tillvalet **-x**, kan ett volymnamn anges så att enbart filer på dessa volymer läses in. Wildcard kan användas när volymnamnet anges om de omges av apostrofer för att hindra shell av avkoda dem. Endast volymnamn som finns i logg-filen kan anges.

-M Sänder meddelande med mail till användaren om något fel uppstår under säkerhetskopieringen.

Tillval för att lägga till, ändra eller lista filer i kommandofilen för *bup*:

-m Anger filer, som skall säkerhetskopieras, då den månatliga kopieringen skall göras (med *bup* **-bm**).

-w Anger filer, som skall säkerhetskopieras, då veckans kopiering skall göras (med *bup* **-bw**).

-d Anger filer, som skall säkerhetskopieras, då den dagliga kopieringen skall göras (med *bup* **-bd**).

-a Anger filer, som skall säkerhetskopieras och sedan raderas, då kopiering för arkivering görs (med *bup* **-ba**). I detta fall raderas även motsvarande rad i bup:s kommandofil då arkivkopieringen görs.

-k Tar bort en tidigare angiven fil från bup:s kommandofil. Namnet på filen/biblioteket skall återges på exakt samma sätt som den ges i kommandofilen, **/usr/lib/bup/cmd**, inklusive eventuella wildcards. Om filer angivits med tillvalet **-r** måste de anges på samma sätt om de skall tas bort.

-r *findargument* Rekursiva parametrar till find. Detta tillval gör att alla följande parametrar på kommandoraden behandlas som parametrar till kommandot find. Parametrarna la-

gras i *bup*:s kommandofil och används av find för att hitta filerna då själva säkerhetskopieringen utförs. 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.

-L Listar köade filer i kommandofilen för *bup*. Dessa filer kommer att kopieras när någon av tillvalen **-b(mwda)** ges.

Tillval för att lista och ändra logg-filen för bup:

-l Listar filinformationen från *bup*:s loggfil. **-l** kan kombineras med **-u** då man vill lista filer enbart med ett specificerat volymnamn. Om filnamn är specificerade som argument kommer endast information om dessa att listas. Filinformationen listas som med kommandot *ls* **-l**. Om även tillvalet **-v** ges, listas också tiden ut då de kopierades. Wildcards kan användas när filnamn och volymnamn (med **-u**) anges för att söka filer i loggfilen. Filnamn som ges utan full sökväg (pathname) anses vara i nuvarande bibliotek.

-e Radera ut alla logg-rader för en viss volym (given med tillvalet **-u**) från bup:s logg-fil.

Tillval för att lista standard volymnamn:

-nm Visar de standardvolymnamn som skall användas vid nästa månatliga säkerhetskopiering. Namnet hämtas från filen **/usr/lib/bup/volumenames**.

-nw Visar de standard volymnamn som skall användas vid nästa veckas säkerhetskopiering. Namnet hämtas från filen **/usr/lib/bup/volumenames**.

-nd Visar de standard volymnamn som skall användas vid nästa dags säkerhetskopiering. Namnet hämts från filen **/usr/lib/bup/volumenames**.

Tillval för att utföra säkerhetskopieringen:

-bm Utför den månatliga säkerhetskopieringen. Alla filer som har angivits med **-m**, **-w** eller **-d** kommer att kopieras till den enhet som anges.

-bw Utför veckokopieringen. Alla filer som har angivits med **-w** eller **-d** kommer att kopieras till den enhet som anges.

-bd Utför den dagliga kopieringen. Alla filer som har angivits med **-d** kommer att kopieras till den enhet som anges.

-ba Utför arkivkopiering till den angivna enheten och **raderar** filerna. Detta påverkar enbart filer som angivits med **-a**. Motsvarande rader med **-a** i *bup*:s kom-

mandofil raderas automatiskt efter det att säkerhetskopieringen är genomförd. Om inget volymnamn angivits (med **-u**) kommer *bup* att skapa ett volymnamn från dagens datum och klockslaget. Detta volymnamn bör anges på volymen för identifiering.

-c Testa volymen efter backup. *bup* använder *tar* och läser volymen efter backup för att kontrollera att kopiering gått bra.

-tk *volymstorlek* Använder nästa argument (volymstorlek) som storleken på volymen i kbyte. Detta sänds vidare till *tar*-kommandot.

-tb *blockfaktor* Använder nästa argument (blockfaktor) som blockfaktor på volymen. Detta sänds vidare till *tar*-kommandot.

Tillval för att läsa tillbaka filer från backupvolymen:

-x Hämtar filer från ett kopieringsmedium på den givna enheten. *bup* skriver ut volymnamnet på det media (band eller diskett) som användaren skall sätta in 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 nuvarande bibliotekets fullständiga sökvägsnamn. Wildcards kan användas när filnamnen specificeras, om de omges av apostrofer för att hindra shell att avkoda dem. *bup* sparar alltid filerna med fulls sökvägsnamn (pathname). Om användaren vill läsa in en fil till ett annat bibliotek för att där kunna undersöka den, kan *tar*-kommandot med tillvalet **-A** användas istället för *bup* **-x**.

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/volumenames	Fil innehållande standardvolymnamn för den månatliga, veckovisa och dagliga säkerhetskopieringen.
/usr/lib/bup/daily	Ex. på shell-procedurer med bup-kommandon för daglig,
/usr/lib/bup/weekly	veckovis och
/usr/lib/bup/monthly	månatlig säkerhetkopiering

EXEMPEL

```
bup -w '*'
```

Detta kommando anger att alla filer i det nuvarande bibliotek skall kopieras vid nästa veckas säkerhetskopiering.

```
bup -m -r -name \*.c
```

Detta kommando anger att alla *.c-filer i det gällande biblioteket skall kopieras vid nästa månatliga säkerhetskopiering. Wildcard-tecknet '*' citeras här från shell med \.

```
bup -bw -M -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/volumenames**. Eventuellt fel som uppstår kommer att rapporteras med *mail* till användaren.

```
bup -ba -v -tk 720 /dev/mf0
```

Detta kommando kopierar alla filer som angivits med *bup* **-a** och raderar sedan ut dem ur systemet. Eftersom tillvalet **-v** har givits, skrivs all *tar*-info ut under kopieringen. Kopieringen sker till en diskett med storleken 720 kbyte.

```
bup -x /dev/st0 file1 file2 file3
```

Detta kommando återladdar filerna file1 file2 och file3 till det aktuella biblioteket. *bup* visar det riktiga volymnamnet på det media (band eller diskett), som skall sättas in innan filerna laddas. För filer, som inte skall tillhöra nuvarande bibliotek, måste fullständiga sökvägsnamn anges.

NAMN

cancel - avbryt begärd utskrift till LP skrivare.

SYNTAX

cancel [-V] [id ...] [skrivare ...]

FUNKTION

cancel tar bort en utskriftsbegäran som gjorts med kommandot *lp*. Kommandots argument kan antingen vara ett id-nummer returnerat av *lp*, eller ett logiskt skrivarnamn. *lpstat* ger en förteckning. Om id-nummer på en utskriftsbegäran anges så avbryts även en pågående utskrift. Om skrivarnamn anges avbryts pågående utskrift på skrivaren ifråga. I båda fallen fortsätter skrivaren/skrivarna att skriva ut nästa tillgängliga utskriftsbegäran ur kön.

När id-nummer anges, kan *cancel* även ta bort utskriftsbegäran till fjärranslutna skrivare via ett nätverk.

Ytterliga information ges i beskrivningen av kommandot *lp*.

TILLVAL

-V	Skriver ut versionsnumret på kommandot *cancel*

FILER

/usr/spool/lp/*

Se *lpadmin* för en fullständig fil-lista.

HÄNVISNING

lp(1), enable(1), lpstat(1), accept(1M), lpadmin(1M), lpsched(1M), reject(1M)

FELMEDDELANDEN

Inga

NAMN

cat(concatenate) - skriver ut filer

SYNTAX

cat [[-V] -s] [-u] [-v [-t] [-e]] filnamn ...

FUNKTION

Kommandot *cat* läser filer som specifierats med filnamn och skriver ut dem på standard output i den ordning som angetts. Om en fil inte existerar visas ett felmeddelande på skärmen och *cat* avbryts.

Kommadot cat läser från standard input om inga filer anges eller om ett bindestreck (-) anges som filnamn.

Med *cat* kan man:

- skapa nya filer, en i taget
- sammanfoga gamla filer
- lägga till en eller flera filer till en gammal fil. Tillägget sker i slutet av den gamla filen.
- skriva över ett helt nytt innehåll i en gammal fil eller lägga till ett nytt innehåll i slutet av denna.
- skapa en fil med ett helt nytt innehåll från standard input, skriva över en gammal fil eller lägga till i slutet av en gammal fil. Alla tillägg gjorda via standard input måste avslutas med CTRL-D och detta tecken måste ges i första positionen på en ny rad.

TILLVAL

-V	Skriver ut versionsnumret på kommandot *cat*
-u	Med tillvalet **-u** skrivs data in direkt i den specificerade filen. Om man inte använder **-u** kommer data skriven via standard input att lagras i en buffer på 512 bytes. Data skrivs in i filen bara när denna buffer är full.
-s	Med tillvalet **-s** skrivs inga varningar ut för filer som inte kan läsas.
-v	Med tillvalet **-v**, kommer ej skrivbara tecken att skrivas synliga, dock inte TAB, NL och FF. Styrtecken kommer att skrivas som ^X (CTRL-X) och tecknet DEL kommer att skrivas som ^?. Om tecknet har bit 8 satt skrivs den som M-x, där x är motsvarande tecken med de sju minst signifikanta bitarna.
-t	Med tillvalet **-t** tillsammans med **-v**, skrivs TAB-tecknen som ^I. Tillvalet ignoreras om **-v** inte är satt.

-e Med tillvalet **-e** tillsammans med **-v**, skrivs ett $ i slutet på varje rad. Tillvalet ignoreras om **-v** inte är satt.

FILER

Inga

EXEMPEL

Exempel 1:

```
cat
```

Allt som skrivs på tangentbordet kommer att skrivas ut på standard output, dvs normalt terminalen. För att avsluta måste man ange CTRL-D på en ny rad.

Exempel 2:

```
cat >produkt
```

Allt som skrivs på tangenbordet skrivs till filen produkt. Alla rader kommer att skrivas till produkt precis som skrivna på tangentbordet. Ingenting skrivs dock in i filen förrän buffern är full. Ibland är det bättre att skriva direkt till filen. Använd i så fall tillvalet **-u**.

Exempel 3:

```
cat produkt
```

Innehållet i filen produkt skrivs ut på standard output.

Exempel 4:

```
cat produkt descriptor > newprod
```

Slår samman innehållet i file produkt med filen descriptor och skriver ut resultatet i filen newprod. Observera: Innehållet i newprod rensas innan produkt och descriptor skrivs in.

Exempel 5:

```
cat > descrip < newprod
```

Innehållet i filen newprod skrivs till filen descrip. Det gamla innehållet i descrip skrivs över.

Exempel 6:

```
cat produkt descrip >> newprod
```

Sammanlägger innehållet i filen produkt med filen descrip och skriver det i slutet av filen newprod, dvs innehållet från filerna produkt och descrip läggs ihop med innehållet i filen newprod.

HÄNVISNING

cp(1), copy(1), ln(1), mv(1), pr(1), umask(1)

FELMEDDELANDEN

Inga

NAMN

cd - byt till annat bibliotek.

SYNTAX

cd[bibliotek]

FUNKTION

Komandot *cd* byter det bibliotek som man befinner sig i (aktuellt bibliotek) till det som angetts som argument. Utan argument kommer man till det bibliotek som specifierats i shell-variabeln $HOME. Denna sätts automatiskt när man loggar in och kan ändras av användaren.

Om biblioteket som gavs som argument inte utgör ett fullständigt pathname (sökväg), dvs inte börjar med '.', '..', eller '/', söker kommandot efter biblioteket enligt någon sökväg av de som är angivna i shellvariabeln $CDPATH. Olika sökvägar i $CDPATH är skilda åt med ett kolon ':'. Om $CDPATH inte är definierad söker *cd* efter det nya biblioteksnamnet i det aktuella biblioteket.

Användaren av kommandot *cd* måste ha exekveringstillstånd till det nya biblioteket.

Kommandot *cd* är internt i shell och utförs utan att skapa någon ny process. Eventuell omdirigering av I/O ignoreras.

TILLVAL

Inga

HÄNVISNING

sh(1), mkdir(1), chmod(1), pwd(1)

FELMEDDELANDEN

Inga

CD(1)

NAMN

chgrp - ändrar gruppidentitet för filer

SYNTAX

chgrp [-V] grupp fil ...

FUNKTION

Kommandot *chgrp* ändrar grupp-identiteten på en eller flera filer.

Gruppen kan vara antingen ett gruppnamn eller ett numeriskt grupp-ID. Båda finns i grupp-identifikationsfilen **/etc/group**. Den numeriska gruppens identifikation finns också i password-filen, **/etc/passwd**.

Observera: Bara super-user och ägare får ändra gruppidentiteten för en fil.

TILLVAL

-V Skriver ut versionsnumret på kommandot.

FILER

/etc/group
/etc/passwd

EXEMPEL

Exempel 1:

```
chgrp gh postf brevgh brevgg
```

Grupp-id för filerna postf, brevgh och brevgg ändras till gh, dvs gruppen gh får använda de specifierade filerna, enligt de åtkomsttillstånd för gruppen som gäller för filerna. Jämför kommandot *chmod*.

Exempel 2:

```
chgrp 20 postkl postad brevdt
```

Grupp-identiteten för filerna postkl, postad and brevdt ändras till 20, dvs gruppen 20 får använda de specifierade filerna.

HÄNVISNING

chmod(1), chown(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Om *chgrp* används av någon annan än super-user, rensar kommandot alltid 'Set-user-ID'-biten och 'Set-group-ID'-biten för filerna. Jämför kommandot *chmod*.

En numerisk grupp identifierare för en fil behöver inte definieras i filen **/etc/group** eller tillhöra någon särskild användare.

NAMN

chmod (change mode) - ändra åtkomsttillstånd

SYNTAX

chmod [-V] mod filnamn ...

FUNKTION

Med detta kommando ändras åtkomsttillstånden (moderna) för en eller flera filer eller bibliotek. Endast ägaren av filen eller super-user kan göra sådana ändringar.

Tillstånden för en användaren (=ägaren), gruppen såväl som för övriga användare får ändras oberoende av varandra.

Ändringarna kan göras på två olika sätt, i ett absolut eller symboliskt mod.

Parametern 'mode' anges som:

```
oktalt nummer              (absolut)
```

eller

```
<vem><vad><vilka>(,... )  (symboliskt)
```

I symbolisk form är mode uppbyggt enligt nedan:

<vem>

a	alla. Om inte <vem> anges på kommandoraden gäller "alla" som standard.
u	user, dvs ägaren av filen.
g	group, dvs övriga användare i gruppen som äger filen.
o	others, dvs alla övriga.

<vad>

+	addera tillstånd utan att ändra något annat för <vem>.
-	tag bort tillstånd, utan att ändra något annat för <vem>.
=	sätt tillstånd för <vem>. Gäller inte 'set-user-ID' och 'set-group-ID' om inte 'u' eller 'g' är uttryckligen angivna som <vem> (Se nedan).

<vilka>

r	read, dvs tillstånd att läsa filen eller söka filnamn i biblioteket.
w	write, dvs tillstånd att skriva till filen eller att skapa eller ändra filer i biblioteket.
x	execute, dvs tillstånd att använda en fil som ett kommando eller att komma åt ett bibliotek (söka, läsa eller skriva i filerna däri).
s	'set-user-ID' eller 'set-group-ID' bitar. Användar-ID och/eller grupp-ID sätts till filens ägare respektive

grupp när filen används som ett kommando. Observera: När <vem> är satt till u, sätts set-user-ID biten, och när <vem> är satt till g, sätts set-group-ID biten. För att sätta 'set-group-ID' mod måste användarens grupp motsvara filens grupp om man inte är super-user.

t 'sticky bit'. Ändras bara när <vem> är 'u'. Denna mod ignoreras i D-NIX. Alla kommandon i D-NIX är, om utrymmet inte krävs för andra program, kvar i minnet efter avslutning, för snabb återstart.

u Sätt tillstånden lika nuvarande fil/biblioteksägartillstånd.

g Sätt tillstånden lika nuvarande fil/biblioteks gruptillstånd.

o Sätt tillstånden lika nuvarande fil/biblioteks

tillstånd för alla övriga.

- Inga tillstånd. Ett bindestreck som ignoreras

kan läggas in. Detta möjliggör ett format motsvarande listformatet i kommandot *ls* **-l**. Exempel:

```
chmod a=r-x, file
```

none Inga tillstånd (eller ett bindestreck) angivet efter <vad> '=' tar bort alla tillstånd för den aktuella <vem>. 's' moden rensas endast om <vem> uttryckligen innehåller u och/eller g.

I absolut form består mode av oktala siffror och är uppbyggd enligt nedan:

4000 Sätt användar-ID till ID för filens ägare, om filen används som ett kommando.

2000 Sätt grupp-ID till filens grupp-ID om filen används som ett kommando.

1000 Sätt 'sticky bit' när filen används som ett komando.

0400 Sätt läs-tillstånd för ägaren (user).

0200 Sätt skriv-tillstånd för ägaren (user).

0100 Sätt tillstånd att använda filen som ett kommando, eller tillstånd att använda biblioteket för ägaren (user).

0040 Sätt läs-tillstånd för gruppen.

0020 Sätt skriv-tillstånd för gruppen.

0010 Sätt tillstånd att använda filen som ett kommando, eller tillstånd att använda biblioteket för gruppen.

0004 Sätt läs-tillstånd för alla övriga.

0002 Sätt skriv-tillstånd för alla övriga.

0001 Sätt tillstånd att använda filen som ett kommando, eller tillstånd att använda biblioteket för alla övriga.

TILLVAL

-V Skriver ut versionsnumret för kommandot.

FILER

Inga

EXEMPEL

Exempel 1:

```
chmod +x fil
```

Lägg till att fil kan användas som kommando av ägaren, gruppen och alla övriga.

Exempel 2:

```
chmod u+r fil
```

Lägg till läs-tillstånd i filen för ägaren.

Exempel 3:

```
chmod go-w,u+w fil
```

Ta bort skriv-tillstånd i filen, för gruppen och alla övriga, samt lägg till skriv-tillstånd för ägaren.

Exempel 4:

```
chmod 0777 fil
```

Sätt läs/skriv/exekverings-tillstånd i fil för alla, dvs för ägaren, gruppen och alla övriga.

Exempel 5:

```
chmod 0600 fil
```

Sätt läs/skriv-tillstånd i fil för ägaren.

HÄNVISNING

umask(1), ls(1), chown(1), chgrp(1), find(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

När kommandot *chown* eller *chgrp* används av någon som inte är 'super-user' rensas moden 'Set-user-ID' och 'Set-group-ID'

Tillståndet 'sticky-bit' har ingen funktion i D-NIX eftersom operativsystemet alltid behåller program i minnet även efter att programmet avslutats, så länge minnet inte behövs för andra program.

Varning: Om en mod ändras med ett oktalt nummer, ange aldrig mer än 4 oktala siffror, eftersom detta också ändrar filtypen (om användaren har tillstånd som super-user). Jämför tillvalet **-perm** i beskrivningen av kommandot *find*.

NAMN

chown - ändra ägare till filer

SYNTAX

chown [-V] ägare fil ... ...

FUNKTION

Med kommandot *chown* ändras ägaren till filen/filerna. Ägaren är antingen ett login-namn eller en numeriskt användar-ID båda finns i filen password, **/etc/passwd**.

Observera: Endast super-user eller ägaren själv kan ändra ägardefinition till filen.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *chown*

FILER

/etc/passwd

EXEMPEL

Exempel 1:

```
chown rp postf
```

Ägar-identifikationen till filen **postf** kommer att ändras till **rp**.

Exempel 2:

```
chown 108 beda oskar
```

Användaren med användar-ID 108 blir ny ägare till filerna **beda** och **oskar**.

Exempel 3:

```
chown gh *
```

Ägar-identifikationen för alla filer i aktuellt bibliotek ändras till **gh**.

HÄNVISNING

chmod(1), chgrp(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

När *chown* används av någon annan än super-user rensar kommandot alltid eventuella 'Set-user-ID' och 'Set-group-ID' bitar för filerna. Jämför kommandot *chmod*.

Ett numeriskt användar-ID, som angivits som ägare till en fil, behöver inte vara definierat i filen **/etc/passwd**.

NAMN

cmp (compare) - jämför två filer.

SYNTAX

cmp [-V] [-l] [-s] fil1 fil2 [pos1 [pos2]]

FUNKTION

cmp jämför två filer. Är de olika, visas teckenpositionen i filen och och radnummer för första skillnaden. Med tillvalet **-l** visas alla tecken som är olika. Om fil1 sätts till **-**, används standard input. Om en fil är lika med början av den andra rapporteras detta.

Om en eller båda av agumenten pos1 och pos2 anges (oktalt om de inleds med 0, annars decimalt), börjar jämförelsen med tecken nummer pos1 i fil1 och med tecken nummer pos2 i fil2.

TILLVAL

-V	Skriver ut versionsnumret på kommandot *cmp*
-l	Visar teckenpositionen (decimalt) och skiljaktiga tecken (oktalt) för varje skillnad.
-s	Visar ingenting. Ger endast en utgångsstatus. 0 för identiska filer, 1 för skiljaktiga filer och 2 om en fil fattas eller inte kan läsas.

FILER

Inga

HÄNVISNING

diff(1), comm(1), diff3(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Detta kommando bör endast användas för att jämföra binärfiler. Använd *diff* för att jämföra textfiler.

NAMN

copy - kopierar grupper av filer.

SYNTAX

copy [-alnomrdvtuV] namn ... dest

copy [-a -l -n -o -m -r -ad -v -t -u -V] namn .. dest

FUNKTION

Innehållet i ett eller flera bibliotek kopieras till ett annat bibliotek. Hela filsystem kan kopieras eftersom bibliotek och specialfiler skapas vid behov. De bibliotek och specialfiler som skapas på detta sätt, får samma åtkomsttillstånd och flaggor som sina original. De bibliotek som redan existerar på destinationen (dest) behåller sina åtkomsttillstånd och sina flaggor.

Observera att kopiering kan ske från flera bibliotek på en gång. Resultatet är precis detsamma som vid kopiering av ett bibliotek i taget.

Argumentet namn kan vara en fil, ett bibliotek eller en specialfil men den måste existera. Om namn inte är ett bibliotek kommer resultatet av *copy* att vara precis detsamma som vid användning av kommandot *cp*.

Destinationen, dest, måste antingen vara en fil eller ett bibliotek som skiljer sig från namn. Om både namn och dest är bibliotek kommer kommandot *copy* att kopiera en fil i taget till destinationsbiblioteket enligt de tillval som anges.

Om specialfiler i **/dev** (enheter) kopieras, skapas specialfilerna men ingen data från filerna kopieras.

TILLVAL

-V	Skriver ut versionsnumret av kommandot *copy*
-a	Innan kopieringen utförs ställs en fråga, denna måste besvaras med y för att kopiering ska ske. Tillvalet **-ad** sätts automatiskt när **-a** anges.
-l	Kopiering sker endast i de fall där länkar inte kan användas, dvs då filerna inte ligger i samma filsystem. Specialfiler och bibliotek kan inte länkas och kopieras därför alltid.
-n	Destinationsfilen måste vara ny. Om så inte är fallet, kan ingen kopiering ske. Tillvalet **-n** är möjlig bara för filer, inte för bibliotek. För specialfiler är tillvalet **-n** alltid satt.
-o	Detta tillval får endast en super-user använda. När tillvalet sätts kommer filens ägar- och gruppidentifikation att ändras till samma som originalfilens. Om tillvalet inte sätts kommer ägaren att vara den användare som gav *copy* kommandot om inte filen redan existerar. I så fall kommer ägare/grupp ID's att vara samma som i den gamla filen.

-m	Alla kopierade filer kommer att få samma åtkomst- och modifieringstid som originalfilen. Om tillvalet inte är satt kommer modifieringstiden att sättas till den tid då kopieringen utfördes.
-r	Alla bibliotek kommer att undersökas för filer som ska kopieras. Om tillvalet inte är satt kommer inga bibliotek att undersökas och bara de filer som angetts på kommandoraden blir då kopierade (om inte tillvalet -a medger val).
-ad	När ett bibliotek påträffas, får användaren en fråga om tillvalet **-r** skall anses vara satt, dvs skall det bibliotek som påträffas undersökas. Om frågan inte besvaras med ett y ignoreras biblioteket.
-v	När man väljer detta tillval kommer namnen på alla filer som kopieras att skrivas ut på terminalen.
-t	Bevarar trädstrukturen hos de filer/bibliotek som kopieras.
-u	Om både namn och dest finns som filer och namn är äldre än dest, sker ingen kopiering. Endast modifieringstiden testas.

FILER

Inga

EXEMPEL

Exempel 1:

```
copy -a /ds/from /ds/to
```

Kopiera filer från biblioteket **/ds/from** till biblioteket **/ds/to**. Innan varje fil kopieras, ställs en fråga. Om denna fråga besvaras med y som första bokstav, kommer filen att kopieras, annars kopieras den inte.

Exempel 2:

```
copy -n /ds/alfa /ds/beta
```

De filer som finns båda biblioteken **/ds/alfa** och **/ds/beta** kopieras inte. Alla andra filer i **/ds/alfa** kopieras till **/ds/beta**.

Exempel 3:

```
copy -r * tmp
```

Om **tmp** inte existerar kommer biblioteket **tmp** att skapas. Därefter kommer alla filer/bibliotek som finns i nuvarande bibliotek, valda med *, att kopieras. När ett bibliotek påträffas kommer dess innehåll att kopieras till **tmp**, men inget underbibliotek kommer att skapas för innehållet.

Exempel 4:

```
copy -rt * tmp1
```

Om **tmp1** inte existerar kommer först ett bibliotek **tmp1** att skapas. Därefter kommer alla filer/bibliotek som finns i nuvarande bibliotek, valda med *, att kopieras. **tmp1** kommer att bli en exakt kopia av nuvarande biblio-

tek med undantag av de filer som börjar med en punkt '.', vilka inte väljs med *.

Exempel 5:

```
copy -van /mf0 .
```

Kopierar filer från en mountad 5 1/4 tum diskett till nuvarande bibliotek. På skärmen ges frågor och en lista på de filer som kopieras. Filer som redan finns kopieras inte. **Observera** punkten '.' på kommando-raden som markerar nuvarande bibliotek.

HÄNVISNING

cp(1), mv(1)

FELMEDDELANDEN

Inga

COPY(1)

NAMN

cp - Kopierar filer eller bibliotek

SYNTAX

cp [-V] filnamn1 filnamn2

cp [-V] filnamn1 filnamn2 ... bibliotek

FUNKTION

Med syntax variant 1, skapar detta kommando en kopia, filnamn2, av filnamn1. Filen filnamn1 kommer inte att påverkas av kopieringen. Om filnamn2 redan existerar kommer dess innehåll att raderas innan innehållet i filnamn1 kopieras. I detta fall behåller filnamn2 sina gamla tillstånd och ägar-identitet.

Om filnamn2 inte existerar skapas det och filen kommer att få samma tillstånd som filnamn1, men ägar-ID och grupp-ID kommer att bli samma som för användaren som ger kommandot *cp*.

Med syntax variant 2, kopieras en eller flera filer med sina originalnamn till det specifierade biblioteket. Detta bibliotek måste existera.

Det går inte att kopiera från en fil till sig själv.

TILLVAL

-V	Skriver ut versionsnumret på kommandot *cp*

FILER

Inga

EXEMPEL

Exempel 1:

```
cp /ds/add /ds/beta
```

Om filen **beta** redan existerar kommer dess innehåll att raderas innan filen **add** kopieras. Skulle däremot filen **beta** inte existera kommer denna fil att först skapas innan en kopiering kan ske. I båda fallen kommer innehållet i filerna att vara exakt lika efter kopieringen.

Exempel 2:

```
cp add data repeat reader /addera
```

Filerna kommer att kopieras till biblioteket **/addera** och filerna kommer att behålla sina originalnamn.

Exempel 3:

```
cp /gre/chapt* /hty/kap21
```

Alla filer i underbiblioteket **/gre** som börjar på **chapt** kommer att kopieras till underbiblioteket **/hty/kap21**.

HÄNVISNING

mv(1), copy(1), cat(1)

FELMEDDELANDEN

Inga

NAMN

cpio - Kopiera filarkiv, in och ut

SYNTAX

cpio -o [acBvVR]

cpio -i [BcdmrtuvfsSbVA] mönster

cpio -p [adlmruvVRA] bibliotek

FUNKTION

cpio **-o** (copy out) läser standard input för att få en lista över filer (pathname) och kopierar därefter dessa filer till standard output tillsammans med pathnamn och statusinformation. Data fylls ut till närmaste 512-byte gräns.

cpio **-i** (copy in) läser filer från standard input. Dessa antas vara skapade av ett tidigare *cpio* **-o**. Bara filer vilkas namn stämmer överens med 'mönster' läses. Mönster ges efter samma regler som gäller i *sh*. I mönster kommer meta-tecknen ?, * och [...] även att matcha tecknet /. Flera mönster kan anges.
Observera: Metatecken måste citeras (med \ eller apostrofer) för att hindra att de tolkas av shell. Mönstret * gäller som standardvärde, dvs alla filer kommer att läsas.

De lästa filerna kommer att skapas och kopieras till aktuellt biblioteksträd enligt tillvalen nedan. Tillstånden för filen kommer att bli desamma som gällde vid föregående *cpio* **-o**. Ägare/grupp kommer att vara den nuvarande användaren om inte denna är super-user. I så fall behålls den gamla ägar/grupp specifikationen i *cpio* **-o**.

cpio **-p** (pass) läser en filnamnslista (pathname) från standard input för att få en lista på filer som ska kopieras och återskapas i destinationsbiblioteket enligt tillvalen nedan.

TILLVAL

-o	Kopiera ut. Se ovan.
-i	Kopiera in. Se ovan.
-p	Passera. Se ovan.
-a	Återställ åtkomsttiden på infilerna när de kopierats.
-B	In/utdata delas upp i block om 5120 byte per record. Detta tillval gäller ej tillvalet **-p** och är bara meningsfull med data som kommer till eller från en fysisk enhet med variabel block-storlek. Standard block-storlek är 512 bytes.
-d	Bibliotek skapas vid behov.
-c	Skriver header-information i filarkivet med ASCII-tecken för att få bästa översättning mellan olika system.

-r	Döper om filer interaktivt. Om användaren ger en radmatning, hoppas filen över.
-t	Skriver ut en innehållsförteckning över indata. Ingen fil skapas.
-u	Kopiera alltid. Normalt kopieras inte en gammal fil över en nyare med samma namn.
-v	Verbose; en lista med alla filnamn skrivs ut. Tillsammans med tillvalet **-t** har listan ett format liknande kommandot *ls -l*.
-l	Länka filer istället för att kopiera om det är möjligt. Kan bara användas med tillvalet **-p**.
-m	Behåller modifieringstiden för filer. Detta tillval fungerar inte för biblioteket som kopieras.
-f	Kopiera alla filer utom de i mönster.
-s	Kasta om (swap) bytes. Används bara med tillvalet **-i**.
-S	Kasta om (swap) halva ord. Används bara med **-i**.
-b	Kasta om (swap) både bytes och halva ord. Används bara med **-i**.
-V	Skriver ut versionsnummer på programmet *cpio*.
-R	Rekursiv kopiering av bibliotek. Används bara med tillvalen **-p** eller **-o**.
-A	När filer kopieras in från ett filarkiv, tas root tecknet (/) bort från filnamnen. Detta möjliggör återställning av filer i ett temporärt bibliotek även om de sparades med fullständiga filnamn (pathnamn).

FILER

Inga

EXEMPEL

Exempel 1:

```
ls | cpio -o >/dev/st0
```

Detta exempel kopierar innehållet i det aktuella biblioteket till ett bandarkiv.

Exempel 2:

```
find . -depth -print | cpio -oB >/dev/st0
```

Detta exempel kopierar en bibliotekshierarki. Exemplet kan skrivas effektivare som:

```
find . -depth -cpio /dev/st0
```

HÄNVISNING

find(1), ls(1), tar(1)

FELMEDDELANDEN

Felmeddelanden ges och utgångsstatus blir skild från noll vid fel.

ANMÄRKNING

Ett filnamn (pathnamn) kan bara bestå av upp till 128 tecken. Om det finns för många unika länkade filer kan programmet till slut inte hålla reda på dem och information om länkarna tappas bort. Bara super-user kan kopiera specialfiler.

CPIO(1)

NAMN

cron - tidsstyrning av processer

SYNTAX

/etc/cron [-V]

FUNKTION

cron utför kommandon vid specificerade datum och tider. Tidtabellsstyrda kommandon kan specificeras enligt instruktionerna i *crontab*filer. Användaren kan aktivera sin egen *crontab*-fil med kommandot *crontab*. Kommandon som bara ska utföras en gång kan aktiveras via kommandot *at*. Eftersom processen *cron* aldrig avslutas skall den endast startas en gång. Det bästa är att köra *cron* från initierings-processen med filen **/etc/rc.**

cron undersöker bara *crontab*-filer och kommandofiler från *at* vid första uppstart och när en fil ändras. Detta reducerar overhead genom att undersökningar av nya och ändrade filer inte behöver göras regelbundet.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *cron*

FILER

/usr/lib/cron	bibliotek för cron-filer
/usr/lib/cron/log	log-fil
/usr/spool/cron	temporära filer

HÄNVISNING

at(1), crontab(1), sh(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

En listning av allt som utförts med *cron* finns i **/usr/lib/cron/log**.

NAMN

crontab - hanterar en användares crontab-fil

SYNTAX

crontab [-V] [fil]

crontab -r

crontab -l

crontab -a

FUNKTION

Kommandot *crontab* kopierar den angivna filen eller standard input om ingen fil har specificerats, till ett bibliotek som innehåller alla användares *crontab*-filer.

Tillvalet **-r** tar bort användarens *crontab* från biblioteket. *crontab* **-l** listar användarens crontab-fil. *crontab* **-a** lägger till den angivna filen eller standard input till filen (append).

En användare får använda *crontab* om hans namn finns i filen **/usr/lib/cron/cron.allow**. Om den filen inte existerar undersöks filen **/usr/lib/cron/cron.deny** för att avgöra om användaren ska nekas åtkomst till *crontab*. Om ingen av dessa filer finns får endast **root** initiera ett uppdrag. Om endast **deny**-filen existerar men den är tom, tillåts global åtkomst. Filerna **allow/deny** innehåller ett namn per rad.

En *crontab*-fil innehåller rader med vardera sex fält. Fälten avgränsas med mellanslag eller tabbar. De fem första är heltalsmönster som specificerar följande:

```
minut (0-59)
timme (0-23)
datum (1-31)
månad (1-12)
veckodag (0-6 med 0=Söndag).
```

Varje mönster kan vara antingen en asterisk (betyder alla legala värden), eller en lista med element avgränsade med kommatecken. Ett element kan vara ett nummer, eller två nummer separerade med minustecken (betyder fr o m - t o m). Notera att specificering av dagar kan göras i två olika fält (dag i månaden eller veckodag). Om båda specificeras så gäller båda. Exempel:

```
0 0 1,15 * 1
```

betyder att kommandot körs den 1:a och 15:de i varje månad, likaså varje måndag. Om man ska specificera dagar i endast ett fält så ska det andra fältet sättas till *. Exempel:

```
0 0 * * 1
```

(kör kommandot bara på måndagar).

Det sjätte fältet på en rad i filen *crontab* är en sträng som exekveras med shell vid de specificerade tiderna. Ett procent-tecken i detta fält (som ej föregås av \) översättes till ett new-line tecken. Bara första raden (till teck-

net % eller slutet på raden) i en kommandofil bearbetas av shell. De andra raderna är tillgängliga för kommandot som standard input.

Shell anropas från användarens $HOME-bibliotek med värdet av argument 0 lika med *sh*. En användare som vill exekvera sin **.profile** måste uttryckligen ange det i filen *crontab*. *cron* ger en standardmiljö för varje shell och definierar HOME, LOGNAME, SHELL(=/bin/sh), och PATH(=:/bin:/usr/bin:/usr/lbin).

Observera: En användare måste komma ihåg att omdirigera standard output och standard error från ingående kommandon! Om detta ej sker skickas all genererad utdata och felmeddelande till användaren via *mail*.

TILLVAL

-V	Skriver versionsnumret för kommandot *crontab*.
-l	Listar användarens *crontab*-fil.
-r	Tar bort användarens *crontab*-fil.

FILER

/usr/lib/cron	systemets cron-bibliotek
/usr/spool/cron/crontabs	spool area
/usr/lib/cron/log	log-fil
/usr/lib/cron/cron.allow	lista på godkända användare
/usr/lib/cron/cron.deny	lista på vägrade användare

HÄNVISNING

sh(1), cron(1M)

FELMEDDELANDEN

Inga

NAMN

cu - Ansluta terminal till externt system

SYNTAX

cu [-V] [-s *baud*] [-l *enhet*] [-h] [-t] [-d] [-o | -e] [-n] telnr
cu [-V] [-s *baud*] [-h] [-d] [-o | -e] -l *enhet*
cu [-V] [-h] [-d] [-o | -e] systemnamn

FUNKTION

Kommandot *cu* används för att koppla terminalen till ett främmande system för interaktiv kommunikaton med möjlighet att överföra textfiler.

Med automatisk uppringning läser *cu* uppringningssekvenserna ur systemfiler i biblioteket **/usr/lib/uucp**. Med enbart systemnamnet angivet, enligt tredje syntaxen ovan, tas även telefonnummer, baudrate och lämplig linje ur systemfilerna. Parametrar kan istället anges direkt, **-l** för val av den enhet som används som port mot den andra maskinen, t ex via ett modem, och **-s** för baudrate. Även i detta fall krävs att linjen är definierad i systemfilen **Devices** och, med automatisk uppringning, även i **Dialers**.

För att kunna ringa ut på en linje som också är öppen för inloggning finns en låsfil i **/usr/spool/locks** som skapas då linjen är upptagen. Denna låsfil används även av *getty, kermit, uucp* (1) m fl program.

Efter att uppkopplingen gjorts kommer *cu* att delas upp i två processer. Den ena fungerar som sändare och den andra som mottagare och de arbetar oberoende av varandra. Sändaren läser standard input och skriver till porten, mottagaren läser från porten och skriver till standard output. Medan detta pågår uppträder *cu* fullständigt transparent för användaren, med undantag för följande interna kommandon.

Om man skriver ~ (tilde) som första tecken på raden efter ett föregående radslut-tecken kan vissa i sändaren inbyggda kommandon aktiveras. På samma sätt kan funktionen i mottagaren styras genom att en inkommande rad börjar med ~.

I sändaren avkodas följande kommandon, efter ett överfört radslutstecken (RETURN-tangenten).

~.	Avslutar körningen med *cu*.
~!	Startar ett temporärt shell i egna datorn.
~!cmd	Utför kommandot *cmd* på eget system via *sh* **-c**.
~$cmd	Exekverar *cmd* på eget system och sänder utdata till den andra sidan.
~%cd	Byt bibliotek i det egna systemet. Observera att detta **ej** kan göras med kommandot *~!cd*, eftersom detta kommando kör i ett "subshell".

~%take ***remote*** **[*local*]**

Kopierar filen **remote** på fjärrsystemet till **local** på egna systemet. Om inte parametern **local** finns, används filnamnet **remote** också i det egna systemet. Kommandot kräver att två Unix-system används och att både *echo* och *cat* finns tillgängliga. Om TAB-tecken inte ska expanderas skall *'stty tabs"* ges efter inloggning till den andra maskinen.

~%put ***local*** **[*remote*]**

Kopierar filen **local** på den egna maskinen till **remote** på den andra. Om **remote** inte anges, används filnamnet **local** på båda systemen. Det fordras att två Unix-system används och att *stty* och *cat* finns tillgängliga samt att tecknen 'erase' och 'kill' (enligt *stty*) är kompatibla i båda systemen.

~~*rad* Sänder raden som börjar med **~*rad*** till det andra systemet, användbart om en ny *cu* har startats på det andra systemet och ett ~ kommando skall ges till denna.

~%break Sänder BREAK-tecken på linjen till den andra datorn. Det kortare **~%b** kan även användas.

~%debug Kopplar av och på debug funktionen. Med debug kommer fler meddelanden att visas på terminalen. Det kortare **~%d** kan även användas. Motsvarar tillvalet **-d**.

~t Skriver ut värdet på termio:s strukturvariabler för användarens terminal.

~l Skriver ut värdet på termio:s strukturvariabler för fjärrkommunikationslinjen.

~%nostop Kopplar av och på XON/XOFF protokollet (DC3/DC1). Kommandot växlar mellan användning eller ej av XON/XOFF under kommunikationen. Standard är med XON/XOFF.

Mottagaren känner igen följande sekvenser från den andra datorn efter ett radsluts-tecken. Dessa kommandon genereras automatiskt när kommandot **~%take** utförs.

~>:*filnamn* Omdirigerar indata från linjen till *filnamn* istället för att sända den till standard output.

~>>:*filnamn* Som ovan men kopieringen sker till slutet av *filnamn*.

~> Återgång till standard output.

De värden som gäller vid start av *cu*, om inget anges, är:

```
Hastighet = Any (vilken som helst som finns tillgänglig)
Ingen paritet
Xon/Xoff signalering
```

XON/XOFF kan ändras med kommandot enligt ovan, hastighet och paritet med tillval vid start.

TILLVAL

-V Skriver ut versionsnumret för kommandot *cu*

-s *baud* Anger överföringshastigheten (baudrate). Standardvärdet är **Any**. Överföringshastigheten kan vara upp till 38400 baud och skall överensstämma med en rad i filen **Devices**. Tillvalet **-s** ignoreras om ett systemnamn anges.

-l *enhet* Anger vilket enhetsnamn som ska användas för kommunikationen. Detta tillval kan användas för att specificera en annan kommunikationsväg än det som systemet väljer i första hand med angiven baudrate (**-s**) Enhetsnamnet kan ges fullständigt, exempelvis **/dev/tty05**, men endast sista delen av namnet används i filen **Devices**.

När tillvalet **-l** används utan tillvalet **-s**, hämtas hastigheten för kommunikation från filen **/usr/lib/uucp/Devices**. När tillvalen **-l** och **-s** används tillsammans, kommer *cu* i alla fall att söka i filen **Devices** för att kontrollera att den begärda hastigheten finns tillgänglig på enheten. Om den finns tillgänglig kommer uppkoppling att ske, annars kommer ett felmeddelande att skrivas ut och uppkopplingen kommer inte att ske. Den specificerade enheten är normalt en direktkopplad asynkron kommunikationskanal när **-l** och/eller **-s** anges, och då bör strängen *telnr* ej anges som parameter och motsvarande rad i filen **Devices** skall innehålla ordet **direct** i fältet för modemtyp.

Om den specificerade enheten är försedd med autouppringning måste ett telefonnummer ges som parameter *telnr*. Tillvalet **-l** kan inte användas om ett systemnamn givits.

-n Begär att användaren ska ange telefonnumret interaktivt, istället för att ta det från kommandoraden.

-h Emulerar lokalt eko, dvs systemet understödjer fjärrdatorer som används i halv duplex mod.

-t Används vid uppringning till en ASCII-terminal som är försedd med autosvar. Omvandlingen mellan vagnretur och vagnretur/radframmatning sköts automatiskt.

-d Diagnostik skrivs till standard error. Användbart för test medan systemfilerna ändras för att införa nya fjärrdatorer eller enheter.

-e, -o Jämn respektive udda paritet genereras på sända data. (e = jämn, o = udda). Standard är ingen paritet.

Ett av de följande alternativ kan användas som sista argument på kommandoraden för *cu*.

telnr När *cu* används tillsammans med automatuppringning till ett system som inte finns som systemnamn i systemfilerna, skall detta argument vara telefon-numret. Nummersträngen kan innehålla '=' där modemet ska vänta på rington och '-' där modemet ska göra en kort paus (ca 4 sekunder).

systemnamn Ett systemnamn kan användas istället för telefon-nummer. Då kommer *cu* att använda en direkt linje eller ett telefonnummer angiven i filen **/usr/lib/uucp/Systems**. *cu* kommer att prova de telefonnummer eller direktlinjer som specificerats för 'systemnamn' i filen **Systems** till dess att anknytning skett eller alla möjligheter är provade. I detta fall är även överföringshastigheten specificerad i samma fil. Även telefonnummer specificeras där, eventuellt med en alfabetisk förkortning som är en referens till filen **Dialcodes**. Om systemnamnet anges på kommandoraden kan inte tillvalen -n, -l eller -s användas.

FILER

/usr/spool/locks/LCK..ttyXX	Låsfiler
/usr/lib/uucp/Systems	Systemnamn och referenser till Devices, eventuellt telefonnummer med förkortning som refererar till Dialcodes.
/usr/lib/uucp/Devices	Linjer, överföringshastigheter och eventuellt referens till modemtyp i Dialers.
/usr/lib/uucp/Dialers	Uppringningssekvenser för olika modem.
/usr/lib/uucp/Dialcodes	Förkortningar i telefonnummer.
/usr/lib/uucp/Sysfiles	Valfri fil där separata systemfiler kan anges för *cu* och *uucp*.

Kommandot *cu* använder normalt samma systemfiler som UUCP nätverksprogram. Nedan ges en kort beskrivning med exempel för de systemfiler som används. Se **System Administration, UUCP nätverk.** för mer detaljer.

FILEN /usr/lib/uucp/Devices

Filen **/usr/lib/uucp/Devices** innehåller förbindelsernas typ samt information om de är fasta förbindelser eller om de är kopplade till ett automatuppringande modem. Det kan finnas mer än en rad som beskriver samma typ av förbindelse. Dessa rader kommer då att bilda en grupp. Om en förbindelse är upptagen försöker *cu* med nästa förbindelse i gruppen i inskriven ordning. Formatet på raderna i filen är

```
Enhetstyp Linje Linje2 Klass Uppringare
```

Följande rader illustrerar olika typer av förbindelser. Observera att fältet *Line2* innehåller ett bindestreck (-).

```
Direct   tty03 - 9600 direct
ACU      tty05 - 1200 hayes
```

Den första raden är för en fast punkt-till-punkt-förbindelse med överföringshastigheten 9600 baud mellan två system.

Den andra raden är för en förbindelse med ett 1200 baud uppringande modem med hayes modemprotokoll. Uppringarsträngen "hayes" refererar till motsvarande rad i filen **Dialers**. Om modemet (t.ex Hayes 1200) kan användas för både 300 och 1200 baud kan en annan rad med samma utseende som den första (förutom hastigheten) skapas. Två rader för olika över- föringshastigheter gör det möjligt att ringa ut till noder med modem för någondera av hastigheterna.

FILEN /usr/lib/uucp/Systems

Filen **/usr/lib/uucp/Systems** innehåller rader, som representerar system vilka kan anropas av bl a *cu*. Mer än en förbindelse kan finnas tillgänglig till ett visst system. Dessa extra förbindelser är då alternativa förbindelsevägar vilka kommer att användas i inskriven ordning tills en uppkoppling kan ske. Formatet på raderna i filen är:

```
Systemnamn Tid Enhetstyp Klass Telefon Loginsekvens
```

Fälten *Enhetstyp* och *Klass* används för att hitta tillåtna enheter i filen **Devices**. Typiska rader i en **Systems**-fil kan vara enligt nedan. *cu* ignorerar fältet *Tid* och *loginsekvensen* i slutet.

```
pp00 Any Direct 9600 tty03 ogin:--ogin:--ogin: nuucp
pp10 Any ACU 1200 04876543 ogin:--ogin:--ogin: nuucp
```

Det första exemplet visar en fast uppkopplad förbindelse. med 9600 baud.

Andra exemplet visar användning av automatuppringande modem med 1200 baud, med telefonnumret angivet. I motsvarande rad i filen **Devices** (med ACU och 1200) anges vilken modemtyp som används i fältet *uppringare*, för vilken styrkoderna finns i filen **Dialers**.

FILEN /usr/lib/uucp/Dialcodes

Denna fil innehåller förkortningarna som används i *Telefon*-fältet i filen **Systems**. Raderna har följande format, där 'abb' anger en alfabetisk förkortningssträng och 'dial-seq' anger ett nummer, eventuellt med koderna = och -.

```
abb dial-seq
```

Filen kan t ex innehålla följande rader:

```
firmab 111
telepro 01234567
```

I det första exemplet skulle en rad med **firmab2222** i fältet *Telefon* i filen **Systems**, skicka 1112222 till det automatuppringande modemet. I det andra exemplet skulle en rad med **telepro** i fältet *Telefon* skicka 01234567 till modemet.

FILEN /usr/lib/uucp/Dialers

Dialers är en modemprotokollfil som används för att beskriva hur man ringer med automatuppringande modem. Dessa protokoll används av *cu* för att ringa ut. Fältet *Uppringare* i filen **Devices** hänvisar till en rad i denna fil.

En modemprotokollfil innehåller rader enligt följande:

```
uppringare substitution förväntad sänd ... ... ...
```

Exempel på en rad ur filen **Dialers**:

```
hayes =,-, "" \dAT\r\c OK\r \EATDT\T\r\d\d\c CONNECT
```

Den andra strängen (=,-,) anger konverteringar av = och - i telefonnumret för just detta modem. Från tredje strängen förväntas varje udda sträng komma från modemet och varje jämn sträng sänds till modemet.

Det finns ett antal specialsträngar som kan användas för speciella ändamål då de ingår som delar av en sänd-sträng.

\p	Paus på mellan 1/4 och 1/2 sekund.
\d	Fördröjning på ca två sekunder innan fler tecken skickas eller läses.
\D	Telefonnummer utan **Dialcodes**-översättning.
\T	Telefonnummer med **Dialcodes**-översättning.
\K	Sätter in en BREAK.
\E	Slår på echo kontroll.
\e	Slår av echo kontroll
\c	Om det är slutet på en sträng undertrycks ny-rad-tecknet, som normalt skickas.
\r	Skickar ett vagnreturtecken (carriage return).
\n	Skickar tecknet för ny-rad (linefeed).
\xxx	Skickar ett tecken vars oktala representation är xxx (t.ex \012, vilket är linefeed).

EXEMPEL

För att ringa upp ett system med telefonnummer 08 - 12 34 56 med hastigheten 1200 baud anges följande, varvid en lämplig linje söks i filen **Devices**. Med exemplet ovan på **Devices**, används tty05 och ett modemprotokoll av typen hayes. Om hastigheten inte var angivet skulle 'Any' användas.

```
cu -s1200 08123456
```

För att logga in på ett direktkopplat system anges följande, varvid motsvarande enhet i **Devices** måste vara markerad som enhetstypen **Direkt**.

```
cu -l /dev/tty03
```

or

```
cu -l tty03
```

För att logga in på systemet pp10, som finns på en rad i filen Systems, anges enbart dess systemnamn, varvid resterande information tas från **Systems**, **Devices** och eventuellt **Dialers**.

```
cu pp10
```

HÄNVISNING

cat(1), echo(1), getty(1M), kermit(1), stty(1), uname(1), uucp(1C)

FELMEDDELANDEN

Felmeddelanden ges i klartext till standard error.

ANMÄRKNING

Om *cu* ska kunna ringa ut på en tty-port som normalt används för login till systemet, måste parametern **-u** ges till *getty* i filen **/etc/inittab** för att tvinga getty att använda en låsfil.

Modem som används måste vara byglade så att bärvågssignalen (DCD) inte ges då modemet är passivt, eftersom *getty* kommer att aktiveras (och skapa en låsfil) så snart DCD kommer från modemet.

Filer överförs utan någon kontroll av överförda data. Överför aldrig filer som har cu-kommandon (med ~) eller som har icke-skrivbara tecken.

CU(1C)

NAMN

cut - Klipper ut selekterade fält från varje rad i en fil.

SYNTAX

cut [-V] -*clista* [fil1 fil2 ...]

cut [-V] -*flista* [-d*char*] [-s] [fil1 fil2 ...]

FUNKTION

Används för att klippa ut kolumner från en tabell eller fält från varje rad i en fil. I databas-termen betyder det implementation av projektionen till en relation. Fälten som specificeras med lista kan ha fasta längder, t ex teckenpositioner som på ett hålkort (tillvalet **-c**) eller varierade längder från rad till rad markerade med fältavgränsnings-tecken som TAB (tillval **-f**). *cut* kan användas som ett filter. Om inga filer anges används standard input.

Använd *grep* för att göra horisontella "utklipp" (baserat på innehållet) av en fil.

TILLVAL

Tillvalen har följande betydelse:

-V Skriver ut versionsnumret för kommandot *cut*

list En lista av komma-separerade heltals-fältnummer (i stigande ordning), eventuellt med - (bindestreck) för att ange ett antal fält eller ett område med fält. Exempel:

```
1,4,7; 1-3,8; -5,10
```

(förkortning för 1-5,10); eller 3- (förkortning för tredje till och med sista fältet).

-*clista* Lista som följer **-c** (utan mellanslag) anger teckenpositioner Exempel:

```
-c1-72
```

tar endast med de första 72 tecknen på varje rad).

-*flista* Listan som följer **-f** är en lista på fält som antages vara separerade i filen med avgränsningstecken (se **-d**). Exempel:

```
-f1,7
```

kopierar endast det första och sjunde fältet, inklusive avgränsningstecknet. Rader utan fältavgränsare kommer att tas med i sin helhet (användbart för underrubriker i tabeller), om inte **-s** har specificerats.

-d*char* Tecknet som följer **-d** är fältavgränsare (endast med tillvalet **-f**). Standard är *tab*. Mellanslag och andra tecken med speciell betydelse för shell måste citeras med \ eller apostrof.

-s Skriver inte ut rader utan fältavgränsare vid användandet av tillvalet **-f**. Om inget angivits, skrivs rader utan avgränsare ut orörda.

Antingen **-c** eller **-f** tillvalet måste specificeras.

FILER

Inga

EXEMPEL

```
cut -d: -f1,5 /etc/passwd
```

listar användarnamn och beskrivning ur **/etc/passwd.**

```
name='who am i | cut -f1 -d" "'
```

sätter variabeln name till aktuellt login namn.

HÄNVISNING

grep(1), sort(1)

FELMEDDELANDEN

```
line too long
```

Raden är för lång. En rad får ej innehålla mer än 1023 tecken eller fält.

```
bad list for c/f option
```

Tillvalet **-c** eller **-f** saknas eller felaktigt specificerad lista. Inget felmeddelande visas om en rad har ett mindre antal fält än vad som behövs för listan.

```
no fields
```

Listan är tom.

NAMN

date - skriver eller sätter datum och tid.

SYNTAX

date [-V] [-u] [mmddttmm [åå]] [+format]

date [-V] [-u] [ååmmddttmm[.ss]] [+format]

date -B ååmmddttmm[.ss] [+format]

FUNKTION

För att skriva ut aktuellt datum och/eller tid, ges kommandot utan datum/tids-argument. En formatsträng som föregås av '+', anger utskriftsformatet, annars används standardformatet såsom framgår av exempel 1 nedan. Utskriften sker till standard output.

För att ändra systemets datum/tid, ges ett argument. Endast en superuser har tillstånd att göra detta. Innan ändring av tid görs bör systemet tas ner till D-NIX enanvändarnivå med kommandot *shutdown*. Datum/ tids-argumentet kan skrivas på det ena eller det andra sättet nedan, varvid minst timme och minut skall anges. Tomma fält ändras inte.

```
Antingen: ((mm)dd)ttmm(åå)
eller: (((åå)mm)dd)ttmm(.ss)
```

Internt använder systemet standard GMT tid (Greenwich Mean Time), men kommandon som *date* eller *ls* använder alltid lokal tid. Skillnaden till lokal tid definieras i shellvariabeln TZ och laddas automatiskt från filen **/etc/timezone** vid inloggning. TZ definierar också textsträngen för tidszonen, vilken kan ses via kommandot *date* (%z format). Om inte TZ definierats, används "Eastern Standard Time", dvs enligt tidszonen i New York. Med tillvalet **-u**, skriver *date* tiden i GMT.

För att ändra tidszonen för sommar respektive vinter, ändras tidsfältet och textsträngen i filen **/etc/timezone**. Normalt anges MET-1 för svensk vintertid och MDT-2 för svensk sommartid. I de fall sommartid enligt 'US daylight time' skall användas, aktiveras detta om en andra textsträng anges efter tidsfältet i TZ. Denna andra sträng används som textsträng vid sommartid och medför automatiskt en timme extra tidsförskjutning enligt reglerna för 'US daylight time'.

Stöd finns för olika länders vecko- och månadsnamn. Värdet av ramvariabeln ENVIRONMENT (se *environ*(5)) avgör vilket språk som skall användas. Namnen på månader och veckodagar i ett visst språk tas från vissa strängar i språkets fil i **/lib/cftime**-biblioteket (se *cftime*(4)).

Efter att ha satt datum och tid kommer *date* att visa nya datumet i det format som angetts i ramvariabeln CFTIME (se *environ*(5)).

För att ändra batteriklockan används tillvalet **-B**. Datum/tid skall då ges som lokal tid i fullständig form enligt ovan. Endast super-user får ändra klockan. Systemtiden ändras samtidigt och systemtiden initieras genom batteriklockan varje gång systemet startas.

Normalt är det inte nödvändigt att ändra batteriklockan i DS90 datorerna eftersom den är initierad vid leveransen. Varje gång systemtiden ändras (utan **-B**) beräknas en korrektionskonstant för batteriklockan som gör att denna går i takt med systemtiden vid uppstart även om batteriklockan inte skulle vara perfekt inställd. För att få tillfredställande noggrannhet bör därför systemklockan ställas ytterligare en gång efter några månader, sedan ändring gjorts i batteriklockan.

FÖRKLARINGAR TILL ARGUMENTFÄLTEN:

åå = år, två sista siffrorna i året. Standard nuvarande år.
mm = månad, månadsnummer (1..12), Standard nuvarande månad.
dd = datum, datum (1..31). Standard nuvarande datum.
tt = timme, (24-timmars system).
mm = minuter, (0..59).
ss = sekunder, (0..59). Standard 0.

FORMAT

Formatsträngen kontrollerar utskriften. Alla tecken utom "%" skrivs ut som de är från formatsträngen. Om mellanslag skall skrivas ut måste formatsträngen ha en apostrof ('...') såväl i början som i slutet av strängen. En sekvens som har "%" åtföljt av något av följande tecken i listan nedan ersätts med motsvarande värde vid utskrift. *date* avslutas alltid med en ny rad.

Sekvensen *date:s* ekvivalenter

%%	%
%a	Förkortad veckodag - Sun till Sat.
%A	Veckodag
%b	Förkortad veckodag - Jan till Dec.
%B	Månad
%d	Datum - 01 till 31.
%D	Datum med formatet - mm/dd/åå.
%e	Dag i månad - 1 till 31, om ett tecken föregås detta av en blank.
%g	Tidszon-text med formatet - GMT+/-tt.mm
%G	Använd GMT-tid. %G skall föregå andra format.
%h	Förkortad månad - Jan till Dec (samma som %b).
%H	Timme - 00 till 23.
%I	Timme 01 - 12
%j	Julianskt datum (Dagsnummer för året) - 000 till 366
%L	Använd lokal tid. %L skall föregå andra format.
%m	Månad - 01 till 12.
%M	Minut - 00 till 59.

%n	Ny rad.
%p	Sträng som innehåller ante-meridiem eller post-meridiem indikator (som standard, AM eller PM)
%r	Tid med formatet - tt:mm:ss AM/PM
%R	Tid som tt:mm
%S	Sekund - 00 till 59.
%t	TAB-tecken, tabulerar till nästa position.
%T	Tid med formatet - tt:mm:ss.
%U	Årets veckonummer (söndag som första dag i vecken) - 01 till 52.
%w	Veckodag - 0 (Söndag) till 6 (Lördag).
%W	Årets veckonummer (måndag som första dag i veckan) - 01 till 52.
%x	Landsspecifikt datumformat.
%X	Landsspecifikt tidsformat.T
%y	Två sista siffrorna i årtalet - 00 till 99.
%Y	År som fyra siffror, ex: 1989
%z	Textsträng för lokal tidszon.
%Z	Namn på tidszon

%L är standard om inget anges. %G eller %L skall anges före tidsformaten.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *date*
-u	Sätter eller läser tiden i GMT.
-B	Sätter batteriklockan i systemet. Utan argument sker ingen ändring, men följande visas: Systemklockan, batteriklockan, tiden för senaste ändring av batteriklockan samt aktuell korrektionskonstant.

FILER

/etc/timezone
/etc/profile
/lib/cftime

EXEMPEL

Exempel 1:

```
date
```

Nuvarande datum och tid skrivs ut i standard format, vilket blir för lokal tid som nedan:

```
Sun Oct 05 21:07:00 MET 1986
```

Exempel 2:

```
date 8607121840
```

Systemets datum och tid blir ställda till 86-07-12 18:40.

Exempel 3:

```
date 1540
```

Kommadot ställer bara tiden och lämnar datum oförändrat.

Exempel 4:

```
date '+ Lokal tid: %L %D %T %g %n GMT: %G %D %T'
```

Både lokal tid och GMT-tiden blir utskrivna på två rader:

```
Lokal tid: 12/07/86 18:41:36 GMT+60:00
GMT: 12/07/86 17:41:36
```

Exempel 5:

```
date -B 8607121740.00
```

Detta kommando ställer batteriklockan med lokal tid som argument.

HÄNVISNING

ls(1)

FELMEDDELANDEN

Ett felmeddelande visas om någon annan än super-user försöker ändra datum eller om argumenten inte är korrekta.

ANMÄRKNING

Förkortningarna för veckodagar och månadsnamn är på engelska och med inledande stor bokstav.

NAMN

dd - Kopierar och omvandlar (konverterar) en fil.

SYNTAX

dd [-V] [tillval=värde] ...

FUNKTION

dd kopierar angiven indatafil till angiven utdatafil med eventuella konverteringar. Som standard används standard input och standard output. Kommandot *dd* är speciellt lämpat för att läsa och skriva på råa fysiska enheter eftersom full kapacitet kan nås genom att blockstorleken kan alterneras för både indata och utdata.

Efter exekvering, visar *dd* antalet fullständigt och delvis fyllda indatablock och utdatablock. Jämför exempel 5 nedan.

TILLVAL

-V	Skriver ut versionsnumret av kommandot *dd*
if=file	Filnamn för indata; annars läses från standard input.
of=file	Filnamn för utdata; annars skrivs till standard output.
ibs=n	Blockstorlek n bytes för indata (standard är 512).
obs=n	Blockstorlek n bytes för utdata (standard är 512)
bs=n	Sätter blockstorlek på både in- och utdata och har högre prioritet än både ibs och obs. Speciellt lämpligt om ingen konvertering skall ske, eftersom ingen intern kopia behövs.
cbs=n	Bufferstorlek för omvandling.
skip=n	Hoppar över n indatablock innan kopieringen påbörjas.
seek=n	Söker n block från början av utdata-filen innan kopieringen påbörjas.
count=n	Kopierar bara n indatablock.
conv=ascii	Omvandlar EBCDIC till ASCII.
conv=ascswe	Omvandlar EBCDIC till Svenska ASCII (med äöåÄÅÖ).
conv=ebcdic	Omvandlar ASCII till EBCDIC.
conv=ibm	Något annorlunda konvertering från ASCII till EBCDIC.
conv=lcase	Omvandlar bokstäver till gemena (Ej äöåÄÖÅ).
conv=ucase	Omvandlar bokstäver till versaler (Ej äöåÄÖÅ).
conv=swab	Skiftar varje byte-par (**swap**).
conv=noerror	Avbryter inte arbetet vid ett fel.
conv=sync	Blankutfyller alla indatablock till **ibs**.
conv='...,'	Flera omvandlingar separerade med komma-tecken.

Där storlekar anges, förväntas ett antal bytes. Ett tal kan avslutas med k, b eller w för att ange multiplikation med 1024, 512 eller med storleken av ett heltalsord i systemet. I DS90-system är ett ord normalt 4 bytes. Ett talpar kan separeras med x för att ange en produkt.

cbs=n används enbart om omvandling till ASCII eller EBCDIC angivits. I det första fallet placeras **cbs** tecken i omvandlingsbuffern, omvandlas till ASCII, avslutande blanktecken tas bort och NL läggs till innan raden skickas som utdata. I det senare fallet läses ASCII-tecken in i omvandlingsbuffern, omvandlas till EBCDIC och blanktecken läggs till för att fylla ett utdatablock av **cbs** storlek.

FILER

Inga

EXEMPEL

Exempel 1:

```
dd if=myfile of=newfile
```

Kopierar myfile till newfile. Jämför kommandot *cp*.

Exempel 2:

```
dd if=myfile of=newfile conv=ucase
```

Kopierar myfile till newfile och omvandlar alla gemena till versaler.

Exempel 3:

```
dd if=/dev/mf0 of=xfile count=1970
```

Kopierar från enheten **/dev/mf0** (en diskett) till filen xfile. count indikerar att man skall läsa 1970 block med standardvärdet 512 bytes/block.

Exempel 4:

```
dd if=xfile of=/dev/mf0
```

Kopierar filen xfile till enheten **/dev/mf0**. Hela filen till filslut kopieras.

Exempel 5:

```
dd if=/dev/mf0 of=xfile ibs=256 count=1970
```

Läser 1970 st block från enheten **/dev/mf0** med blockstorlek 256 bytes/block till filen xfile med block storlek 512 bytes/block, vilket är standard blockstorlek för utdata.

dd avslutar med att skriva ut hur många block den läst in och hur många den har skrivit. Observera att antalet skrivna i detta exempel är hälften av de lästa, eftersom blockstorlekarna är olika.

```
1970+0 records in
985+0 records out
```

Exempel 6:

```
dd if=/dev/mt0 of=afile ibs=800 cbs=80 conv=ascii,lcase
```

Kommandot läser in ett EBCDIC band från enheten **/dev/mt0**, i block om tio 80-bytes EBCDIC record per block, till ASCII-filen afile. Alla bokstäver blir omvandlade till små under överföringen och NL-tecken blir inlagda efter varje omvandlat EBCDIC-record.

HÄNVISNING

copy(1), cp(1), tar(1), cpio(1), mount(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

Konverteringen ASCII till EBCDIC, 'ebcdic' är hämtad från standard CACM Nov, 1968, och konverterar alla 256 ASCII-tecken (8-bitar). 'ibm' konverteringen är något annorlunda och liknar den vanliga IBM-konventionen för skrivare.

DD(1M)

NAMN

df - anger kvarvarande ledigt utrymme på skivminnen.

SYNTAX

df [-V] [filsystem ..]

FUNKTION

df skriver ut det lediga diskutrymme och den totala volymstorleken i kbytes på anslutna D-NIX filsystem. Ett eller flera filsystem kan ges som argument (t.ex. **/dev/si0** eller **/dev/mf0**). Om argumentet filsystem icke anges, rapporteras allt fritt utrymme i samtliga filsystem till standard output. Kommandot *df* använder listan på alla anslutna filsystem i filen **/etc/mnttab.**

TILLVAL

-V Skriver ut versionsnummer till kommandot *df.*

FILER

/dev/*
/etc/mnttab
/..systemfiles/bitmap

EXEMPEL

Exempel 1:

```
df
```

Listar ledigt diskutrymme på alla anslutna filsystem i systemet.

Exempel 2:

```
df /dev/mf0
```

Listar fritt diskutrymme på filsystemet på en mountad 5 1/4 tum diskett.

HÄNVISNING

fsck(1M), mount(1M), umount(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

Utrymmet kan listas endast för standard filsystem och inte för filsystem som är anslutna med speciella filhanterare.

Kommandot *df* är systemberoende.

NAMN

devnm - skriver ut namn på fysiska enheter

SYNTAX

/etc/devnm [-V] filer ....

FUNKTION

Skriver ut namn på de fysiska skivminnen där filer finns. Filnamnen skrivs ut efter enhetsnamnen med en rad per fil. Fullt pathname måste anges för filerna på kommandoraden. Enhetsnamnet skrivs ut utan delen '/dev/' i namnet.

/etc/devnm kommandot skapar lämplig utdata att fungera som indata till kommandot **/etc/setmnt**.

TILLVAL

-V Skriver ut versionsnumret för kommandot *devnm*.

FILER

/dev/*

EXEMPEL

Exempel 1:

```
/etc/devnm /usr/jag
```

Raden som skrivs ut kan se ut som nedan, beroende på systemet:

```
si0 /usr/jag
```

HÄNVISNING

setmnt(1M)

FELMEDDELANDEN

Inga

DEVNM(1)

NAMN

dirname - ger biblioteksdelen av ett pathname.

SYNTAX

dirname [-V] sträng

FUNKTION

dirname ger allt utom den sista delen av pathname i strängen och skriver resultatet på standard output. Om pathname bara har en komponent skrivs bara en punkt. Normalt används kommandot *dirname* inom '....' i shell-procedurer.

Det besläktade kommandot *basename* tar bort alla prefix som slutar med /, samt eventuella suffix.

TILLVAL

-V Skriver versionsnummer på *dirname* kommandot.

EXEMPEL

Exempel 1:

```
NAME='dirname /usr/src/cmd/cat.c'
```

Sätter shellvariabeln NAME till **/urs/src/cmd**

Exempel 2:

```
dirname /a/b/c/d
```

Skriver **/a/b/c** på standard output.

Exempel 3:

```
dirname fil.ext
```

Skriver en punkt på standard output.

HÄNVISNING

basename(1), sh(1)

FELMEDDELANDEN

Inga

DIRNAME(1)

NAMN

disable - Desaktiverar LP-skrivare

SYNTAX

disable [-V] [-c] [-r[*orsak*]] skrivare

FUNKTION

Disable passiverar namngivna skrivare, hindrar dem att skriva ut köade utskriftsbegäran, som sänts med kommandot *lp*.

Detta kommando finns förklarat mer i detalj tillsammans med kommandot *enable*.

DISABLE(1)

NAMN

dmacs - skärmorienterad text editor

SYNTAX

dmacs [-Vaervxn] [-*gnum*] [-k*key*] [-s*string*] [@fil] [fil ... ...]

FUNKTION

dmacs är en skärmorienterad texteditor med möjlighet att dela upp en vanlig terminalskärm i flera fönster, samt använda flera olika filbuffertar mellan vilka texter kan flyttas.

dmacs hanterar filer med icke skrivbara tecken genom att visa alla kontrolltecken som ~X där X är motsvarande skrivbart tecken.

Vid start konfigureras *dmacs* med användarkommandon och/eller teckenkonvertering genom att läsa en initieringssekvens från filen **.dmacsrc**. Denna innehåller macrosatser med *dmacs*-kommandon som utförs vid start. Denna fil söks först i användarens hembibliotek, därefter enligt $PATH och i vissa standardbibliotek samt sist i aktuella biblioteket. En användare kan alltså ha sin egen **.dmacsrc** fil i HOME-biblioteket. Shell variabeln $TERM måste definieras och motsvarande fil måste finnas i **/usr/lib/terminfo** för att *dmacs* skall kunna användas.

TILLVAL

-V	Skriver versionsnumret för kommandot *dmacs*.
-a	Gör att *dmacs* utför kommandona i filen **error.cmd** istället för i **.dmacsrc**.
-e	De textfiler som följer på kommandoraden kan ändras (i motsats till tillvalet **-v**, vilken kan ha givits tidigare på kommandoraden).
-r	Begränsar *dmacs*. Tillåter inte utförandet av några kommandon för att läsa/skriva andra filer än de som angivits på kommandoraden. Tillåter inte heller att shell-kommandon anropas.
-v	*d-macs* sätts automatiskt i VIEW mod så att de textfiler som följer på kommandomoden inte kan ändras. Jämför tillvalet **-e.**
-x	Begränsar tangentbordet. XON/XOFF tangenterna (CTRL-S/CTRL-Q) används inte som *dmacs*-kommandon, utan används som handskakningsprotokoll vid kommunikationen. Istället används CTRL-[och CTRL-\.
-n	XON/XOFF-protokollet används inte. Detta är standard, varvid CTRL-S/CTRL-Q kan användas som kommandon.
-g*num*	Markören placeras direkt på rad *num* i filen när *dmacs* startas.
-k*key*	Källkodsfilerna är i krypterad form. *dmacs* använder nyckeln *key* och avkodar texten för att presentera den i

läsbar form. Om ingen *key* specificeras direkt efter **-k**, frågar *dmacs* efter en nyckel utan att eka svaret.

-s*string* När *dmacs* startats, söker den automatiskt efter första sträng *string* i första källkodsfilen och placeras markören där.

Kommandon i den angivna filen utförs istället för i filen **.dmacsrc** innan *dmacs* läser några andra filer. Flera sådana kommandofiler kan anges (med @) i en kommandorad.

FILER

.dmacsrc	I $HOME, enligt $PATH, i /usr/lib/dmacs eller i aktuellt bibliotek.
/usr/lib/dmacs/dmacs.hlp	Fil med hjälptexter som kan ses med kommandot ESC ?.

HÄNVISNING

ed(1), red(1), siv(1)

FELMEDDELANDE

Inga

ANMÄRKNINGAR

Inga

NAMN

du (disk usage) - information om hur skivutrymmet används, uttryckt i kbytes.

SYNTAX

du [-Vsar] [namn ..]

FUNKTION

Kommandot *du* ger storleken på skivutrymme uttryckt i kbytes som aktuellt bibliotek inklusive underbibliotek upptar. Om en fil eller ett bibliotek anges vid namn, kommer kommandot att ge storleken på skivutrymme uttryckt i kbytes som denna fil/bibliotek upptar. Anges inget namn används aktuellt bibliotek. Även filsystem, mountade med speciella filhanterare kan listas med kommandot *du*.

En fil med två eller flera länkar räknas bara en gång.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *du*
-s	Anger den totala storleken i kbytes för alla filer, utan att lista alla underbibliotek separat.
-a	Ger storleken i kbytes för var och en av filerna. För underbibliotek ges normalt endast bibliotekets storlek.
-r	Upphäver den normala undertryckningen av felmeddelanden.

FILER

Inga

EXEMPEL

Exempel 1:

```
du /usr/hty/kap5
```

Anger det totala antalet kbytes skivutrymme som underbiblioteket **/usr/hty/kap5** upptar och listar dess filer. Alla underbibliotek inom **/usr/hty/kap5** blir listade separat, dock blir det totala antalet alltid listat på sista raden.

HÄNVISNING

df(1M), mkfs(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

Om det finns alltför många länkade filer, kommer du att räkna filerna mer än en gång och ett meddelande ges (om **-r** tillvalet används) när detta sker.

Filer med hål får fel storlek.

Storleken på skivminnet inkluderar även det extra utrymme som krävs då filstorleken avrundas uppåt till en multipel av blockstorleken i filsystemet. (Jämför kommandot *mkfs*).

Om tillvalet **-a** inte används och filnamn ges som argument kommer endast filer som är bibliotek att listas, men alla listade filer ingår i totalsumman om även tillvalet **-s** anges.

Detta kommando är systemberoende.

NAMN

echo - ekar det som skrivs som argument

SYNTAX

echo argument ... ...

FUNKTION

Kommandot *echo* skriver ut de angivna argumenten med blanktecken emellan och avslutar med en ny rad. Kommandot kan användas när felmeddelande skall produceras i shellprogram eller för att skriva ett konstant dataflöde till en pipeline.

Kommandot *echo* är internt för shell och exekveras utan att skapa någon ny process. Omdirigering av utdata kan göras. Se anmärkning!

TILLVAL

Inga (Se anmärkning!)

Format

I argumentet kan man ange vissa specialtecken. Dessa föregås av ett "\" enligt nedanstående sekvenser:

Observera att sekvenserna måste placeras mellan enkla apostrofer för att kommandohanteraren inte skall "förstöra" dem.

Sekvens	Utskrift
\b	Tecknet BACKSPACE. Stega bakåt.
\f	Tecknet FORM-FEED. Ny sida.
\n	Ny rad. Radframmatningstecken. Tecknet NL.
\r	Tecknet RETURN. Flyttar markören till början av raden.
\t	Tecknet TAB. Tabulerar.
\v	Tecknet VT. Vertikal tabulering.
\c	Ingen radframmatning ges efter utskrift.
\\	Tecknet \.
\0xxx	xxx representerar noll, ett, två eller tre siffror mellan 0 och 7. *echo* skriver ut det tecken vars ASCII-värde motsvaras av det oktala talet xxx (eller 0, om xxx ej anges).

FILER

Inga

EXEMPEL

Exempel 1:

```
echo Detta visar funktionen hos echo
```

Texten kommer att skrivas ut på skärmen och därefter ställer sig markören på ny rad.

Exempel 2:

```
echo 'Markören kommer att stå kvar på raden \c'
```

Texten skrivs ut på skärmen, med markören placerad direkt efter det skrivna meddelandet. Notera att apostroferna behövs både för att tillåta ö, å, ä liksom för att tillåta specialtecknet \c.

HÄNVISNING

cat(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

För kompatibilitet med tidigare system finns även kommandot */bin/echo*. Detta kommando laddas från en fil och som en process. Det har samma funktion som *echo*, men tillåter dessutom följande tillval. Hela sökvägen */bin/echo* måste anges för att skilja det från det interna kommandot *echo*.

Tillval för ***/bin/echo***:

-V	Skriver ut versionsnummer för kommandot */bin/echo*.
-e	Skriver till standard error. Samma som omdirigering med >&2.
-n	Ingen ny rad efter utskrift. Samma som om texten avslutades med '\c'.
--	Avslutar tillvalet på kommandoraden och tillåter att efterföljande argument kan börja med ett minustecken - utan att tolkas som ett tillval.

NAMN

ed, red - texteditor

SYNTAX

ed [-V] [-s] [-p *sträng*] [-x] [-C] [fil]

red [-V] [-s] [-p *sträng*] [-x] [-C] [fil]

FUNKTION

ÖVERSIKT

ed är en standard textredigerare (texteditor) som startas med kommandot *ed*. När *ed* har startats används *ed*-kommandon för textredigering. Varje kommando har en eller två adresser, som anges med en 'regular expression', i det följande förkortat till RE. Vissa kommandon ges dock utan adress.

Om argument **fil** ges simulerar *ed* ett *e*-kommando (se kommandobeskrivning nedan) på filen ifråga. Detta innebär att filen läses in till en *ed*-buffert för redigering. Ändringar görs på en tillfällig kopia av filen. Originalfilen berörs inte förrän kommandot *w* (write) ges.

red är en begränsad version av *ed*. *red* tillåter endast redigering av filer i aktuellt bibliotek. Dessutom spärras körning av shellkommandon via *!shell command*. Om man försöker gå runt spärren får man ett felmeddelande (restricted shell).

Både *ed* och *red* kan formatera enligt *fspec(4)*. När man anger en formatspecifikation som första rad i en fil och startar *ed* med terminalen i mod *stty* -tabs eller *stty* -tab3 (se *stty*), kommer angivna tabulatorstopp att automatiskt användas vid listning av filen. Om exempelvis första raden innehåller:

```
<:t5,10,15 s72:>
```

kommer TAB-stoppar att sättas i positionerna 5, 10 och 15, och radlängden att sättas till 72. **Observera:** Under inskrivning av text kommer inskrivna tabtecken att sättas i var åttonde position, vilket är standard.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *ed*.
-s	Kopplar bort den teckenräkning som görs av *e*, *r* och *w*-kommandona, diagnostikmeddelanden från *e* och *q*-kommandona, samt !-prompten efter ett *!shell*-kommando.
-	Detta tillval har ersatts av **-s** ovan. Se **ANMÄRKNING**.
-p *sträng*	Ger användaren möjlighet att ange en prompt-sträng. Standard är att ingen prompt används. Ett mellanslag krävs i kommandoraden mellan **-p** och promptsträngen.

-x Kryptering; ej tillgänglig utanför U.S.A. p.g.a. exportrestriktioner; *ed* simulerar kommandot *X* och efterfrågar en kodnyckel. Denna nyckel och krypteringsalgoritmen *crypt(1)* används vid kryptering och dekryptering av text. *X* försöker avgöra om den inlästa texten är krypterad eller ej. Den tillfälliga bufferten krypteras också, men med en annan kodnyckel (se *crypt(1)*). Se **ANMÄRKNING**.

-C Samma som *X* ovan, förutom att *ed* simulerar ett *C*-kommando. Detta *C*-kommando är detsamma som *X*, förutom att den inlästa texten antas redan vara krypterad. Ej tillgängligt utanför U.S.A. p.g.a. exportrestriktioner, se **ANMÄRKNING**.

KOMMANDOSTRUKTUR

Kommandon i *ed* har en enkel och regelbunden struktur: ingen, en eller två adresser, följda av ett en-teckens-kommando, möjligen också följda av kommandoparametrar. Adresserna anger en eller flera rader i bufferten. Varje kommando som kräver adresser har standardadresser, varför adresser ofta kan utelämnas. I allmänhet tillåts endast ett kommando på varje rad.

Vissa kommandon medger inmatning av text. Denna text placeras i avsedd plats i bufferten. Medan *ed* accepterar text sägs den vara i **inmatningsmod**, och inga kommandon kan ges. All inmatning lagras i bufferten. Man går ur inmatningsmod genom att skriva en punkt (.) ensam i början av en rad, följd av RETUR.

RE

ed använder sig av en begränsad form av 'regular expressions', RE. RE används i adresser för att ange rader eller, i vissa kommandon (t.ex. *s*), delar av en rad som på något sätt skall ersättas. En RE definierar en mängd av teckensträngar. En sträng sägs matcha den angivna RE om den uppfyller de villkor som definierar RE. En RE kan konstrueras på följande sätt:

En RE kan ange ett enda tecken, eller vara en sammansättning av sådana, en **hel RE**.

Följande RE matchas av ett enda tecken:

1. Ett vanligt tecken (ej ett av dem som beskrivs under 2. nedan) är en enteckens-RE som matchar sig själv.
2. Tecknet \, (bakvänt snedstreck) följt av ett specialtecken är en enteckens-RE som matchas av detta specialtecken. Specialtecknen är:

a. ., *, [, och \ (punkt, asterisk och tecknen [och \) är alltid specialtecken, utom när de står inom hakparenteser [] (se 4. nedan).

b. ^ (caret eller circumflex) är ett specialtecken endast i **början** av en **hel RE**, eller när det omedelbart följer den vänstra hakparentesen av ett hakparentespar ([]).

c. ¤ (valutasymbol), som är ett specialtecken i slutet av en **hel RE** (se 7.b nedan).

d. Det tecken som används för att avgränsa en **hel RE**, är speciellt för RE:n ifråga (se t.ex. hur snedstreck (/) används i kommandot *-g* nedan).

3. En punkt (.) är en enteckens-RE som matchar varje tecken utom new-line-tecknet.

4. En icke-tom sträng av tecken omgivna av hakparenteser ([]) är en enkel-RE som matchar vilket tecken som helst i denna sträng. Dock, om det första tecknet i strängen är ett circumflex (^), så matchar detta RE vilket tecken som helst utom ett new-line-tecken och de angivna tecknen. Tecknet ^ har denna speciella betydelse endast om det är det första tecknet i strängen. Minustecknet (-) kan användas för att uttrycka en följd av på varandra följande ASCII-tecken ([0-9] är likvärdigt med [0123456789]). Minustecknet har inte denna speciella betydelse om det står först (eller efter ett initialt ^, om sådant också finns) eller sist i en sträng. Höger hakparentes (]) avslutar inte en sträng när det är det första tecknet i strängen (eller efter ett initialt ^, om sådant också finns). Exempel: []a-f] matchar antingen tecknet] eller en av bokstäverna a t.o.m. f. De fyra tecknen under 1. och 2a. ovan representerar sig själva inom en sträng av detta slag.

REGLER FÖR RE

Följande regler kan följas för att konstruera RE från enteckens-RE. (Med RE nedan avses den kompletta, **hela RE**, till skillnad från en enteckens-RE).

1. En enteckens-RE är en RE som matchar allt som enteckens-RE matchar.

2. En enteckens-RE som följs av en asterisk (*) är en RE som matchar **ingen** eller fler förekomster av en enteckens-RE. Om flera strängar i raden matchas, väljs den först påträffade längsta matchande strängen.

3. En enteckens-RE följt av \{m\}, \{m,\}, eller \{m,n\} är en RE som matchar ett visst antal förekomster av enteckens-RE. Värdena m och n måste vara icke-negativa heltal under 256. \{m\} matchar exakt m förekomster; \{m,\} matchar minst m förekomster; och \{m,n\} matchar godtyckligt antal förekomster mellan m och n inklusive. När ett alternativ finns matchar RE så många förekomster som möjligt.

4. En sammanslagning av flera RE är en RE som matchar sammanläggningen av strängar som matchas av varje del av denna RE.

5. En RE omgiven av tecknen \(och \) är en RE som matchar allt som RE mellan \-tecknen matchar.

6. Uttrycket \n, där n är en siffra, matchar samma teckensträngar som matchats av ett tidigare uttryck omgivet av \(och \) tidigare i samma RE. Den delmängd som angetts är den som börjar med den n:te förekomsten av \ (från vänster räknat). Exempel: Uttrycket ^\(.*\)\1¤ matchar en rad som har två upprepade förekomster av samma sträng.

7. Slutligen kan en **hel RE** begränsas till att matcha **början** eller **slutet** av en rad (eller båda).

a. En circumflex (^) i början på en **hel** RE begränsar denna RE till att matcha **början** av raden.

b. En valutasymbol (¤) i slutet på RE begränsar denna RE till att matcha **slutet** av en rad.

Konstruktionen ^RE¤ begränsar en **hel** RE till att matcha hela raden.

En tom RE (t.ex. //) är ekvivalent med den sist påträffade RE. Se även sista stycket före FILER nedan.

ADRESSERING

För att förstå adresseringen i *ed* är det nödvändigt att känna till att det alltid finns en **aktuell rad**, adresserad av tecknet '.' (se 1. nedan). Allmänt gäller att aktuell rad är den rad som senast påverkats av ett kommando. Hur den aktuella raden påverkas behandlas under beskrivningen av varje kommando nedan. Adresser konstrueras enligt följande:

1. Tecknet . (punkt) adresserar aktuell rad.
2. Tecknet ¤ adresserar den sista raden i bufferten.
3. En heltalssiffra n adresserar den n:te raden i bufferten.
4. 'x adresserar raden markerad med tecknet x, vilket måste vara en gemen bokstav (a-z). Rader markeras med kommandot *k* enligt nedan.
5. En RE omgiven av snedstreck (/) adresserar den första rad som påträffas när man söker **framåt** mot slutet av bufferten från raden **efter** aktuell rad, och stannar vid den första rad som innehåller en sträng som matchar RE. Om nödvändigt fortsätter sökningen till början av bufferten och fortsätter till och inkluderar aktuell rad, så att hela bufferten genomsöks. Se även det sista stycket före FILER nedan.
6. En RE omgiven av frågetecken (?) adresserar den första rad som påträffas när man söker **bakåt** mot början av bufferten från raden **före** aktuell rad, och stannar vid den första rad som innehåller en sträng som matchar RE. Om nödvändigt fortsätter sökningen till slutet av bufferten och fortsätter bakåt till och inkluderar aktuell rad, så att hela bufferten genomsöks. Se även det sista stycket före FILER nedan.
7. En adressdel som följs av ett plustecken (+) eller ett minustecken (-), följt av ett decimaltal anger denna adress, plus (eller minus) det angivna antalet rader. Plustecknet kan uteslutas.
8. Om en adress börjar med + eller -, syftar additionen eller subtraktionen på den aktuella raden, t.ex. -5 betyder .-5 (punkt minus fem).
9. Om adressen slutar med + eller -, lägger man till eller tar bort 1 från adressen. Som följd av denna regel och av regel 8 ovan hänvisar adressen - till raden före aktuell rad. (För kompatibilitet med tidigare versioner av *ed* har tecknet ^ i adresser precis samma funktion som -.). En följd av upprepade + eller - (plus- eller minustecken) fungerar kumulativt. Exempel: -- betyder aktuell rad minus 2.
10. För bekvämlighets skull ersätter komma (,) adressparet 1,¤ och semikolon (;) ersätter adress-paret .,¤.

KOMMANDON I *ed*

Kommandon kan kräva ingen, en eller två adresser. Kommandon som inte kräver någon adress behandlar ev. förekomst av adress som ett fel. Kommandon som godtar en eller två adresser använder standardadresser när ett otillräckligt antal adresser anges. Om fler adresser än nödvändigt anges används den/de sista adressen/adresserna.

Adresser avskiljes vanligen från varandra med ett komma (,) eller semikolon (;). I det senare fallet sätts aktuell rad (.) till första adressen, från vilken den andra adressen beräknas. Denna funktion kan användas för att bestämma startraden för framåt- och bakåt- sökning (se regel 5 och 6 ovan). Den andra adressen i en två-adress-sekvens måste i bufferten motsvara en rad som följer efter den rad som anges av den första adressen.

I den lista på *ed*-kommandon som följer visas standardadresser inom parenteser. Parenteserna i beskrivningen är **inte** en del av adressen. De visar bara att givna adresser är standardadresser.

Det är i regel otillåtet att ha mer än ett kommando på en rad. Däremot kan alla kommandon (utom *e*, *f*, *r*, eller *w*) ha suffixet **l**, **n**, eller **p**, i vilket fall den aktuella raden antingen listas, numreras eller skrivs ut. Se nedan under kommandona *l*, *n*, och *p*.

(.)a

<text> Kommandot *append* läser den givna texten och lägger in den efter den adresserade raden; aktuell rad lämnas vid sist infogade rad, eller, om ingen infogning skett, vid den adresserade raden. Adress 0 är tillåtet för detta kommando: det medför att 'inlagd' text placeras i början på bufferten. Maximalt antal tecken som kan ges från en terminal är 255 per rad (inklusive tecknet newline).

(.)c

<text> Kommandot *change* tar bort de adresserade raderna och tar sedan emot text som ersätter dessa rader. Aktuell rad-tecknet lämnas vid sista inmatade rad, eller om sådan ej finns vid första raden som inte tagits bort.

(.,.)d Kommandot *delete* tar bort adresserade rader från bufferten. Raden som följer den sist borttagna raden blir den aktuella raden. Om raderna som togs bort ursprungligen fanns i slutet på bufferten blir den nya sista raden aktuell rad.

e fil Kommandot *edit* tar bort hela innehållet i bufferten och sedan läses filen in. Aktuell rad-tecknet sätts på sista raden i bufferten. Om inget filnamn anges används det sist använda filnamnet om sådant finns (se kommandot *f*). Antalet tecken som lästes in skrivs ut. Namnet fil behålles för att kunna användas som standard filnamn i efterföljande *e*, *r* och *w*-kommandon. Om

fil ersätts med !, förutsättes resten av raden vara ett shell-kommando *sh(1)*, vars utdata skall läsas in. Ett sådant shell-kommando används inte som aktuellt filnamn.

E fil Kommandot ***Edit*** är detsamma som kommandot *e* , men med den skillnaden att *edit* inte kontrollerar om några ändringar har gjorts i bufferten sedan det sista *w*-kommandot användes.

f fil Om **fil** anges ändrar detta filnamns-kommando aktuellt filnamn till **fil**; i annat fall skriver kommandot ut det sist använda filnamnet.

(1,¤)g/RE/kommandolista

Med kommandot ***global*** markeras först varje rad som matchar den aktuella RE. Sedan exekveras för varje sådan rad kommando-listan med aktuell rad-tecknet satt på den rad som markerats. Ett enkel-kommando eller det första kommandot på en lista anges på samma rad som det globala kommandot. Alla rader i en flerrad-slista utom den sista raden måste sluta med \. *a*, *i* och *c*-kommandon med indata är tillåtna. Avslutande av indata-mod med .(punkt) kan uteslutas om det är den sista raden i kommando-listan. En tom kommandolista är lika med kommandot *p*. Kommandona *g*, *G*, *v* och *V* är ej tillåtna i kommandolistan. Se även sista stycket före FILER nedan.

(1,¤)G/RE/ I det interaktiva kommandot ***Global*** sker först en märkning av rad som matchar den givna RE. Sedan skrivs varje sådan rad ut. Aktuell rad-tecknet ändras till att gälla denna rad, och alla andra kommandon (utom kommandona *a*, *c*, *i*, *g*, *G*, *v* och *V*) kan anges och exekveras. Efter exkvering av ett kommando skrivs nästa markerade rad ut, o.s.v. Ett new-line-kommando behandlas som ett noll-kommando. Ett & har till följd att det sist exekverade kommandot inom *G* utförs igen. Observera att kommandon körda med *G*-kommandot kan adressera och påverka alla rader i bufferten. Kommandot *G* kan avslutas med en avbrottssignal (ASCII DEL eller BREAK).

h Kommandot ***help*** ger ett kort felmeddelande som förklarar orsaken till den sista ?-meddelandet.

H Kommandot ***Help*** låter *ed* övergå till en mod där felmeddelanden skrivs ut för alla följande ?-diagnoser. Det förklarar även det föregående ?, om det fanns något. *H* avslutar och påbörjar denna mod alternerande. Det är från början avstängt.

(.)i

<text>

Kommandot *insert* infogar given text före den adresserade raden. Aktuell rad-tecknet (.) lämnas vid den sist infogade raden, eller, om en sådan rad ej finns, vid den adresserade raden. Detta kommando skiljer sig från kommandot *a* bara i placeringen av den inmatade texten. Adress 0 får ej användas med detta kommando. Maximalt kan man ge 255 tecken per rad (inklusive new-line-tecknet) från en terminal.

(.,.+1)j

Kommandot **join** sammanfogar på varandra följande rader genom att ta bort new-line-tecken. Om endast en adress anges utför detta kommando ingenting alls.

(.)kx

Markeringskommandot **k** markerar den adresserade raden med ett tecken (t.ex. **x**), som måste vara en gemen bokstav (a-z). Adressen 'x' adresserar då denna rad. Aktuell rad förblir oförändrad.

(.,.)l

Kommandot *list* skriver ut de adresserade raderna på ett entydigt sätt. Vissa oskrivbara tecken representeras av skrivbara tecken(t.ex. skrivs TAB, BACKSPACE ut som >, respektive <). Alla andra oskrivbara tecken skrivs på oktal form; långa rader delas upp med tecknet \. Kommandot *l* kan användas med andra kommandon med undantag av *e*, *f*, *r*, och *w*.

(.,.)ma

Kommandot *move* flyttar den adresserade raden (raderna) efter raden adresserad med **a**. Adress 0 är tillåten för **a**, och ger upphov till att den adresserade raden (raderna) flyttas till början av filen. Ett fel uppstår om adress **a** ligger inom de flyttade raderna. Aktuell rad-tecknet står kvar vid den senast flyttade raden.

(.,.)n

Kommandot *number* skriver ut de adresserade raderna; varje rad föregås av radnumret och ett TAB-tecken. Aktuell rad-tecknet står kvar vid den senast skrivna raden. Kommandot *n* kan användas med alla kommandon utom *e*, *f*, *r*, och *w*.

(.,.)p

Kommandot *print* skriver ut de adresserade raderna. Aktuell rad-tecknet står kvar på den senast skrivna raden. Kommandot *p* kan användas med alla kommandon utom *e*, *f*, *r*, och *w*. Exempel: *dp* tar bort den aktuella raden och skriver ut den nya aktuella raden.

P

Editorn frågar med ett * efter alla efterföljande kommandon. Kommandot **P** startar och avbryter denna mod alternerande, på och av. Vid start är den av.

q

Med kommandot *quit* avslutas *ed*. *ed* skriver ej den fil den redigerat, men om ändringar har gjorts i filen måste kommandot *q* ges två gånger för att avsluta *ed*.

Q Editorn avslutas **utan** att kolla om ändringar har gjorts i bufferten sedan det sista *w*-kommandot exekverades.

(¤)r fil Kommandot *read* läser in den namngivna filen efter den adresserade raden. Om inget filnamn angetts används det senast använda filnamnet, om sådant finns (se kommandon *e* och *f*). Det senast använda filnamnet ändras inte, såvida inte **fil** är det först angivna filnamnet efter att *ed* har startats. Adress 0 är tillåten för *r*, och innebär att texten från filen lägges in från början av bufferten. Om exekveringen av kommandot går problemfritt skrivs antalet tecken ut. Aktuell rad-tecknet sätts på sist lästa rad. Om **fil** ersätts med **!** anses resten av raden vara ett shell-kommando vars utdata skall läsas in. Till exempel fogar **'¤rlls '** aktuellt bibliotek till slutet av den fil som redigeras. Ett sådant shellkommando används inte som aktuellt filnamn.

(.,.)s/RE/ersättningssträng/ eller

(.,.)s/RE/ersättningssträng/g eller

(.,.)s/RE/ersättningssträng/n n = 1-512

Ersättningskommandot *s* söker i varje adresserad rad efter en förekomst av den angivna RE. På varje rad som matchar ersätts alla (ej överlappande) matchande strängar om den globala ersättningsindikatorn **g** förekommer efter kommandot. Om inte, ersätts endast den första förekomsten av en matchad sträng. Om talet **n** anges efter kommandot ersätts endast den n:te förekomsten på varje adresserad rad. Ett fel anses ha uppstått om ersättningen misslyckas på någon adresserad rad. Alla tecken utom mellanslag och newline-tecknet kan användas istället för / för att avgränsa RE och ersättningssträngen. Aktuell rad-tecknet lämnas kvar på den rad där ersättningen sist gjordes. Se även det sista stycket före FILER nedan.

Om ett **ampersand**-tecken,(&), förekommer i ersättningen ersätts den av den sträng som matchar RE på den aktuella raden. Den speciella betydelsen av & i detta sammanhang kan undertryckas om & föregås av ett \. Mer vanligt är användning av ersättstecknen **\n**, där **n** är ett tal för den text som matchar den n:te 'sub-RE' av en RE omgiven av \(och \). När flera nivåer av sub-RE inom parenteser förekommer, beräknas **n** genom att räkna förekomster av \(, med början från vänster. Om tecknet **%** är det enda tecknet i ersättningen används den ersättning som gällde för föregående ersättningskommando att gälla även för det aktuella. Tecknet % förlorar sin speciella betydelse i en

ersättningssträng som består av flera tecken, eller som föregås av \.

En rad kan delas upp genom att införa ett new-line-tecken på raden. New-line-tecknet i ersättningen kan enbart ges genom att den föregås av ett \. En sådan ersättning kan inte göras i en kommando-lista med *a*, *g* eller *v*.

(.,.)ta Detta kommando fungerar precis som kommandot *m*, med undantag av att en kopia av de adresserade raderna placeras efter adress **a** (som kan vara 0). Aktuell rad-tecknet lämnas kvar på den sista raden av kopian.

u Kommandot *undo* upphäver verkan av det senast använda kommando som ändrade något i bufferten, d.v.s. det senaste av kommandona *a*, *c*, *d*, *g*, *i*, *j*, *m*, *r*, *s*, *t*, *v*, *G*, eller *V*.

(1,¤)v/RE/kommandolista

Detta kommando är samma som det globala kommandot *g*, med den skillnaden att kommandolistan exekveras med aktuell rad initialt satt till varje rad som inte matchar RE.

(1,¤)V/RE/ Detta kommando är samma som det interaktiva globala kommandot *G*, utom att de rader som markerats i första skedet är de som **inte** matchar RE.

(1,¤)w fil Kommandot *write* skriver ut de adresserade raderna i den angivna filen. Om filen inte existerar skapas den med mod 666 (till vilken alla användare har läs- och skrivprivilegier) om inte kommandot *umask* bestämmer annorlunda. Det senast använda filnamnet ändras inte, såvida inte **fil** är det allra första namnet givet efter start av *ed*. Om inget filnamn anges används det senast använda filnamnet, om sådant finns (se kommandona *e* och *f*). Aktuell rad förblir oförändrad. Efter avslutat kommando skrivs antalet tecken ut. Om **fil** ersätts med **!** behandlas resten av raden som om den vore ett shellkommando, vars "standard input" är de adresserade raderna. Ett sådant shell-kommando sparas inte som aktuellt filnamn.

(¤)= Radnumret på den adresserade raden skrivs ut. Aktuell rad ändras inte.

!shell-kommando

Resten av raden efter tecknet **!** sänds till shell (*sh*) som ett kommando. I parametrarna kan tecknet **%** användas och ersätts med det senast använda filnamnet i *ed*. Om ett **!** förekommer som det första tecknet i shell-kommandot ersätts det med texten i föregående shell-

kommando. !! kommer därför att upprepa det senaste shell-kommandot. Om någon expansion utförs kommer denna rad att visas på skärmen. Aktuell rad är oförändrad.

(.+1)<new-line> En adress som står ensam på en rad förorsakar att den adresserade raden skrivs ut. Ett ensamt new-line-tecken är lika med *.+1p*. Detta är användbart när man vill bläddra framåt i bufferten.

ÖVRIGT

Om en avbrotts-signal (ASCII DEL eller BREAK) ges skriver *ed* ut ett ?, och går tillbaka till kommando-nivån.

Några storleksbegränsningar: 512 tecken per rad, 256 tecken per global kommando lista, 64 tecken per filnamn, och 128k tecken i bufferten. Begränsningen på radantalet är beroende av minnesstorleken i systemet: varje rad tar 1 ord (word) förutom textinnehållet.

När *ed* läser en fil tas ASCII NUL-tecken och alla tecken efter den sista NEW-LINE bort. Filer som innehåller tecken som inte finns i ASCII-tabellen (med 8:e biten satt) kan inte redigeras av *ed*.

Om ett begränsningstecken för en RE eller för en ersättningssträng (t.ex. /) skulle vara det sista tecknet före en new-line kan det utelämnas, i vilket fall den adresserade raden skrivs ut. Följande kommando-par är ekvivalenta:

```
s/s1/s2  s/s1/s2/p
g/s1     g/s1/p
?s1      ?s1?
```

FILER

$TMPDIR/e#	Den tillfälliga arbetsfilen placeras i biblioteket **$TMPDIR** (i st.f. i **/usr/tmp**), om ramvariabeln **TMPDIR** är skild från noll. # är processnumret.
usr/tmp/e#	Bibliotek för den tillfälliga arbetsfilen, om **TMPDIR** är noll.
/tmp/e#	Bibliotek för den tillfälliga arbetsfilen, om **TMPDIR** inte existerar eller har värdet noll, och om biblioteket **/usr/tmp** inte existerar.
ed.hup	i denna fil sparas arbetet om terminalen hänger sig på något sätt.

HÄNVISNING

fgrep(1), grep(1), sed(1), sh(1), stty(1), umask(1), vi(1).

fspec(4), regexp(5) i **D-NIX 5.3 Systemadministration.**

FELMEDDELANDEN

? Vid kommando-fel.

?fil Vid en oåtkomlig fil. (använd kommandona *help* eller *Help* för en detaljerad förklaring.)

Om ändringar har gjorts i en buffert sedan det sista *w*-kommandot skrev hela bufferten, varnar *ed* användaren om han försöker radera *ed*'s buffert med kommandona *e* och *q*. Den skriver ut ett **?** och tillåter sedan fortsatt redigering.

Ytterligare ett kommando *e* eller *q* vid detta tillfälle kommer nu att radera bufferten, såvida inte tillvalet **-s** använts vid start.

ANMÄRKNING

Tillvalet **-** har ersatts av **-s** (se *intro(1)*).

Kommandot **!** kan inte exekveras med ett *g* eller *v*-kommando. Kommandot **!** och specialtecknet **!** i kommandona *e*, *r*, och *w* kan inte användas om editorn har startats från ett restriktivt shell (se *sh(1)*).

Sekvensen **\n** i en RE matchar inte ett new-line-tecken.

Om redigerings-kommandon kommer från en kommando-fil (t.ex. genom kommandot *ed* **fil < ed-cmd-fil**), kommer redigeringen att avslutas vid det första misslyckade kommandot i kommandofilen.

Krypteringsalgoritmen för UNIX, version 3.1 är p.g.a. exportrestriktioner ej tillgänglig utanför U.S.A. Algoritmen finns därmed ej heller i D-NIX, version 5.3.

NAMN

errdemon - loggning av felmeddelanden.

SYNTAX

/usr/lib/errdemon [-Vci] [binfile] [ASCII-fil]

FUNKTION

Processen *errdemon* samlar felmeddelanden från operativsystemet genom att läsa specialfilen **/dev/error** och placerar dem i den binära filen **binfile**. Om **binfile** inte anges när *errdemon* startas används **/usr/adm/errfile**. Notera att **binfile** skapas om den inte finns. Om den finns läggs felmeddelanden till och tidigare loggade meddelanden går inte förlorade.

Om parametern asciifile ges, avkodar *errdemon* också felmeddelanden och skriver ut dem i läsbar ASCII-text på ASCII-fil. Om ett bindestreck anges som ASCII-fil, skrivs texten ut på standard output.

errdemon kan också skriva ut felmeddelanden i ASCII-form till huvudkonsolen eller på standard output. Den kan också användas för utskrift i asciiform av tidigare sparade binära fel-loggningsfiler.

Felloggningsprocessen startas normalt i bakgrunden av processen *init* och kan stoppas med *kill*. Endast superuser får starta den. Endast en *errdemon* åt gången får vara aktiv och läsa från enheten **/dev/error**.

TILLVAL

-V	Skriver ut versionsnumret av kommandot *errdemon*
-c	Styr felmeddelanden i ASCII-form till systemets huvudkonsol (**/dev/console**). En ASCII-fil kan också anges.
-i	Läser binära felmeddelanden från den binära filen binfile istället för från **/dev/error**. Skriver ut felmeddelanden i ASCII-form på standard output. (Parametern ASCII-fil ignoreras).

FILER

/dev/error	enhet för läsning av felmeddelanden
/usr/adm/errfile	standard logg-fil för lagring av binära felmeddelanden.

EXEMPEL

```
/usr/lib/errdemon -i /usr/adm/errfile
```

Detta kommando läser de binära felmeddelanden som tidigare sparats i filen **/usr/adm/errfile** och skriver ut dem på standard output.

HÄNVISNING

kill(1), init(1M)

FELMEDDELANDEN

Meddelanden som skapas av *errdemon* är självförklarande.

NAMN

enable, disable - aktivera/deaktivera LP skrivare.

SYNTAX

enable [-V] skrivare ...

disable [-V] [-c] [-r[*orsak*]] skrivare ...

FUNKTION

Enable aktiverar de namngivna skrivarna, möjliggör för dem att skriva utskrifter ur kön från lp-kommandot. Använd *lpstat* för att lista status för skrivare.

Disable deaktiverar angivna skrivare och hindrar utskrift till dessa från kön. Utskrifter som pågår till de angivna skrivarna stoppas, men kommer att skrivas ut i sin helhet på nytt senare till samma skrivare eller till en annan skrivare. Använd *lpstat* för att lista status för skrivare.

TILLVAL

Användbara tillval för *enable* är:

-V	Skriver ut versionsnumret för kommandot *enable*

Användbara tillval för *disable* är:

-V	Skriver versionsnummer för kommandot *disable*
-c	Stoppar alla pågående utskrifter på de angivna skrivare.
-r[*orsak*]	Knyter en orsak till passiveringen. Denna orsak gäller för samtliga skrivare tills ett nytt **-r** tillval ges. Om **-r** tillvalet inte finns eller om **-r** tillvalet angivits utan en orsaks-sträng kommer en standardtext att användas. Orsaken kan listas med *lpstat*.

FILER

/usr/spool/lp/*

HÄNVISNING

accept(1M), lp(1), lpstat(1), lpadmin(1M), lpsched(1M), lpshut(1M), reject(1M)

ANMÄRKNING

Om textsträngen *orsak* innehåller mellanslag, måste den omslutas av apostrof, exempelvis:

```
disable -r 'Detta är orsaken' main.
```

NAMN

env - ändra omgivningsparametrar.

SYNTAX

env [-V] [-] [namn=värde] ... [kommando args]

/bin/env [-V] [-] [namn=värde] ... [kommando args]

FUNKTION

env tar nuvarande omgivning (environment), ändrar den enligt argumenten och startar kommandot med den ändrade omgivningen. Argumenten anges på formen namn=värde och lägges till omgivningen innan kommandot startas. Tillvalet - gör att nuvarande omgivning helt ignoreras så att kommandot utförs med enbart de parametrar som definierats som argument till kommandot *env*.

Om inget kommando anges, skriver *env* ut de omgivningsparametrar som är definierade, ett namn=värde par per rad.

Kommandot */bin/env* kan användas för att direkt starta ett kommando från filen **/etc/passwd** med definierade parametrar.

TILLVAL

-V	skriver ut versionsnumret för kommandot *env*

FILER

Inga

HÄNVISNING

sh(1), exec(1)

FELMEDDELANDEN

Inga

NAMN

eval - substituerar och utför kommandon

SYNTAX

eval argument ....

FUNKTION

Kommandot *eval* används i samband med kommando-substitution när en andra översättning av metatecken skall göras av shell i en substituerad kommandosträng.

Kommandot *eval* är internt för shell och exekveras utan att skapa någon process. Omdirigering av in/utdata är inte tillåten för kommandot *eval* självt, men kan ges till de kommandon som exekveras med *eval*.

TILLVAL

Inga

FILER

Inga

EXEMPEL

```
wg='who | grep'     Definierar en pipeline för ett shellkommando.
$wg kurt            Här kommer inte pipesymbolen att tydas som ett
                    specialtecken.
eval $wg kurt  eval möjliggör avkodning av pipe-symbolen efter
                    parameter-substitutionen.
```

HÄNVISNING

sh(1)

FELMEDDELANDEN

Inga

EVAL(1)

NAMN

exec - exekverar ett kommando istället för nuvarande shell

SYNTAX

exec kommandofil argument .....

FUNKTION

Kommandot *exec* laddar och exekverar ett kommando genom att ersätta ett pågående shell, utan att skapa en ny process.

exec är intern i shell.

In/utdata kan omdirigeras precis på samma sätt som vid exekvering av en kommandofil utan *exec* aktuell shellmiljö kan också ändras om kommandot *exec* ges utan kommandofil som argument. Exempelvis kan andra filer än standard input, output och error på detta sätt öppnas i nuvarande shell. Filnummer 0, 1 och 2 motsvarar de tre standardfilerna.

TILLVAL

Alla tillval angivna efter kommandofilen, kommer att användas av det givna kommandot.

FILER

Inga

EXEMPEL

Exempel 1:

En användares kommandofil (en shellprocedur) med följande innehåll kommer att starta ett basic program utan att skapa ett extra sub-shell för shellproceduren.

```
exec basic /usr/ucb/myprog.bas
```

Exempel 2:

I detta exempel på shellprocedur öppnas en fil med filnummer 4 och frågor läses från den. Standard input (0) har temporärt omdirigerats till den öppnade filen (4) för att varje 'read'-kommando skall kunna läsa en rad i taget. Om standard input omdirigerades direkt till filen för varje gång skulle alltid endast första raden läsas.

```
exec 4<questions     # Open file 'questions'
while read QUESTION <&4
do echo $QUESTION '\c'
read ANSWER
   # do some useful command, depending on $ANSWER.
done
```

HÄNVISNING

sh(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Om *exec* exekveras från tangentbordet från inloggnings-shell eller från en shellprocedur definierat som startprogram i **/etc/passwd**, kommer användaren att loggas ut efter det att det exekverade kommandot avslutats.

NAMN

exit - avsluta shell, eller utloggning om kommandot ges från login shell.

SYNTAX

exit [nn]

FUNKTION

Kommandot *exit* avslutar aktuellt shell. Utgångsstatus blir värdet nn om detta är angivet som argument. Standard utgångsstatus är lika utgångsstatus från sist utförda kommando.

Om kommandot *exit* ges direkt från ett login shell och inte från en shellprocedur, loggas användaren ut.

exit kodas av internt av shell.

TILLVAL

Inga

FILER

Inga

HÄNVISNING

sh(1)

FELMEDDELANDEN

Inga

EXIT(1)

NAMN

export - överför (exporterar) shellvariabler.

SYNTAX

export [NAMN[=sträng] .....]

FUNKTION

Shellvariabler skapas i den shellomgivning (environment) man arbetar i och är dessutom tillgängliga för alla kommandon som startas från denna shell.

Argumentet NAMN är namnet på en shell variabel, definierad som möjlig att exportera och eventuellt given ett nytt värde (sträng). Flera argument kan ges.

export utan argument skriver ut en lista på exporterbara variabler.

Kommandot *export* är internt i shell och exekveras utan att skapa en ny process. Omdirigering av in/utdata är inte tillåten.

TILLVAL

Inga

FILER

Inga

HÄNVISNINGAR

set(1), sh(1).

FELMEDDELANDEN

Inga

EXPORT(1)

NAMN

expr - utvärderar argument som uttryck.

SYNTAX

expr [-V] argument .....

FUNKTION

Argumenten tas som ett uttryck. Efter utvärderingen skrivs resultatet till standard output. Delarna i uttrycket måste separeras med blanktecken. Tecken reserverade i shell måste föregås av "\". Notera att 'noll' (0) i texten nedan anger värdet 0, medan 'null' indikerar en tom sträng. Strängar innehållande blanktecken eller andra speciella tecken exempelvis |, &, ^, >, < och * måste sättas inom apostrofer. Argument med heltalsvärden kan föregås av ett minus-tecken och kan ges i decimal, hexadecimal eller oktal form. Ett numeriskt argument är hexadecimalt om det börjar med 0x eller 0X, oktalt om det börjar med 0, annars anses det vara decimalt. Internt hanteras värden som 32-bitars två-komplementära heltal.

Operatorer och nyckelord listas nedan. Uttryck skall sättas inom apostrofer i shell, eftersom många av tecknen som har särskild betydelse i shell också har särskild betydelse i *expr*.

Listan är i prioritetsordning, med operatorer med samma prioritet grupperade inom paranteser. Parenteser som måste citeras med \ (... \) i shell, kan användas i uttrycken för att ändra ordning på evalueringen.

Utgångsstatus för kommandot *expr* är:

0 Erhållet uttryck är varken 'null' eller 0.

1 Erhållet uttryck är NULL eller 0.

2 Fel i utvärderingen av uttrycket.

uttryck \| uttryck

Operatorn \| returnerar första uttrycket om det inte är NULL eller 0, annars returneras andra uttrycket.

uttryck \^ uttryck

Operatorn \^ (XOR) behandlar endast uttryck som är heltalsvärden och returnerar resultatet efter bit-vis XOR av de två värdena.

uttryck \& uttryck

Denna operator returnerar första uttrycket om inget av dem är NULL eller 0. Annars returneras NULL eller 0.

\~ uttryck Operatorn \~ (NOT) behandlar endast uttryck som är heltalsvärden och returnerar ett-komplementet av uttrycket, dvs ett värde där varje bit är inverterad.

uttryck (==, \>, \>=, \<, \<=, !=) uttryck

Ger resultatet av en aritmetisk jämförelse om båda uttrycken är heltal, annars ges resultatet av en alfabe-

tisk jämförelse. Sant (true) ger strängen "1" och falskt (false) ger strängen "0".

uttryck (+, -) uttryck

Addition eller subtraktion av heltalsvärden.

uttryck (*, /, %, \<\<, \>\>) uttryck

Multiplikation, division, eller restvärde av heltals-division, 'vänster shift' eller 'höger shift' av heltalsvärden. Vid skift anger andra uttrycket antal bitar att skifta.

uttryck : uttryck

Jämförelseoperatorn ':' jämför första uttrycket (strängen) med det andra, vilket måste vara ett korrekt reguljärt uttryck. Vanligtvis returnerar jämförelseoperatorn en siffersträng som är det antal tecken som är lika, eller en delsträng av det första uttrycket om mönstret \(... \) i det reguljära uttrycket använts. Det reguljära uttrycket skall anges mellan apostrofer om det innehåller något av shell's specialtecken.

En korrekt syntax av ett reguljärt uttryck är samma som i kommandot *ed*. Jämförelsen börjar alltid med det första tecknet i det första uttrycket, som om '^' angetts i början av det reguljära uttrycket. Därför är '^' i denna position inte ett specialtecken till *expr*. I det reguljära uttrycket, kan en punkt '.' representera vilket tecken som helst, uttrycket '.*' representerar en sträng av ett godtyckligt antal tecken, tecknet 'v*' representerar en sträng av ett godtyckligt antal av tecknet 'v'.

TILLVAL

-V Ger versionsnumret för kommandot *expr*

FILER

Inga

EXEMPEL

Exempel 1:

```
a='expr $a + 1'
```

Adderar 1 till shellvariabeln a.

Exempel 2:

```
expr 3 \* 0x400 \>\> \( 16 / 2 \)
```

Multiplicerar 3 och 400 Hex, och ger 3072. Dividerar 16 med 2, och ger 8. Skiftar sedan första resultatet 8 bitar till höger och returnerar därefter resultatet 12. (3 kbytes = 0xc00 = 3072).

Exempel 3:

```
expr abcdef : abc
```

Antal jämförda tecken, siffran 3, returneras.

Exempel 4:

```
expr abcdef : '..\(..\)'
```

Substrängen 'cd' returneras eftersom den består av det 3:e och 4:e tecknet från den första strängen.

Exempel 5:

```
expr $d : '.*'
```

Returnerar antalet tecken i $d. Det reguljära uttrycket '.*' motsvarar hela strängen i shell variabeln $d och alla tecken ingår.

Exempel 6:

```
expr $d : '.*/\(.*\)' \| $d
```

Om shellvariabeln antas innehålla en hel sökväg (pathname) med tecknet '/', ger kommandot *expr* filnamnet, vilket är en delsträng av $d och består av sista tecknen i strängen, men bara de som följer det sista '/'-tecknet, dvs filnamnet. Den sista delen av kommandosträngen '\ | $d' läggs till för att ta hand om fallet när $d inte har något '/' tecken. I detta fall ger jämförelse-operationen talet 0 och hela kommandot *expr* returnerar hela $d. Detta är detsamma som att använda kommandot *basename*.

HÄNVISNING

sh(1), ed(1)

FELMEDDELANDEN

Felmeddelanden ges vid syntax-fel och om aritmetiska operationer görs på en icke-numerisk sträng.

ANMÄRKNING

När argumenten har bearbetats med shell kan *expr* bara skilja operatorer och operander åt på deras värden. Om till exempel $a är ett likhetstecken (=) kommer kommandot

```
expr $a = '='
```

att se ut som

```
expr = = =
```

när det sänds till *expr*, där alla tydas som operatorn =. Jämförelser av likhetstecken kan erhållas genom omskrivning av koden och tillägg av ett godtyckligt tecken i båda strängarna:

```
expr A$a = A=
```

Operatorerna \^, \~, \<\< and \>\> är D-NIX-tillägg till normal Unix System V standard.

EXPR(1)

NAMN

false - ger utgångsstatus skilt från noll.

SYNTAX

false

FUNKTION

false gör inget annat än att ge utgångsstatus skild från noll. Motsatsen *true*, gör inget annat än att ge noll som utgångsstatus. *false* används framförallt i shellprocedurer som

```
until false
do
   command
done
```

TILLVAL

Inga

FILER

Inga

HÄNVISNING

sh(1), true(1)

FELMEDDELANDEN

Ger alltid ett utgångsstatus (icke noll) som motsvarar ett kommando som stoppats på grund av fel.

NAMN

fgrep - söker ett mönster i en fil.

SYNTAX

fgrep [-Vvxcilnefpsh] [strängar] [filer]

FUNKTION

Kommandot *fgrep* används till att välja ut rader med angivna strängar från en eller flera filer och skriva ut dessa rader på standard output. Filnamnen skrivs ut i början av varje rad i kommandot endast om mer än en fil är angiven. Utan angivande av filer genomsöks standard input.

Rader i filerna som innehåller någon av angivna strängar väljs och skrivs ut. Om mer än en sträng anges eller om en sträng innehåller mellanslag måste argumentet strängar omges av apostroftecken , '....', och varje sträng skrivas på separat kommandorad. Varje sträng kan bestå av flera ord.

TILLVAL

-V	Skriv ut versionsnumret för kommandot *fgrep*
-v	Skriv endast ut rader som **inte** innehåller någon av strängarna.
-x	Skriv endast ut rader som i sin helhet är lika med någon av strängarna.
-c	Skriv endast ut antalet rader som innehåller någon sträng.
-i	Behandla stora och små bokstäver som lika vid jämförelse med strängarna.
-l	Skriv endast ut namnen på de filer som innehåller någon av stängarna. Inga rader skrivs ut.
-n	Varje utskriven rad föregås av radnumret i filen.
-e *strängar*	Gör det möjligt att ange strängar som börjar med tecknet -. Utan **-e** kommer en sträng som börjar med - att behandlas som om - är ett tillvalstecken.
-b	Varje rad föregås av numret på det block där raden hittades. Detta är ibland användbart när man vill söka ett block med visst innehåll på ett skivminne.
-f *strängfil*	Listan på strängar tas ur filen *strängfil* med en sträng på varje rad. Strängar skall separeras med newlines.

FILER

Inga

EXEMPEL

Exempel 1:

```
fgrep '|' /etc/termcap
```

Listar alla rader som innehåller tecknet | i filen **/etc/termcap.**

Exempel 2:

```
fgrep -n 'abcd
kalle
sträng3' myfile yourfile
```

Listar alla rader som innehåller någon av strängarna abcd, kalle eller sträng3 i de två filerna. Varje funnen rad skrivs ut med filnamnet och radnumret.

HÄNVISNING

grep(1)

FELMEDDELANDEN

Utgångsstatus från *fgrep* är 0 om någon rad hittades med någon av strängarna. Om inga rader hittades är utgångsstatus 1 och om annat fel av något slag registrerats är status 2. Observera att utgångsstatus 2 ges om en angiven fil inte fanns, även om rader hittades i andra filer.

Radlängden begränsas till BUFSIZ tecken; längre rader trunkeras. (BUFSIZ är definierad i **/usr/include/stdio.h** och är normalt 512 tecken).

Om det finns en rad med inbäddade nollor kommer *grep* bara att söka upp till första nollan; om den matchar skrivs hela raden ut.

NAMN

find - söker filer

SYNTAX

find [-V] sökvägs-lista uttryck

FUNKTION

find går rekursivt igenom hela bibliotekshierarkin för varje sökväg (pathname) i listan över sökvägar (d.v.s. en eller flera sökvägar) och söker filer som uppfyller ett booleskt villkor enligt listan nedan.

I beskrivningen används argumentet n som ett decimalt heltal. +n betyder mer än n, -n betyder mindre än n, och n betyder exakt n.

Tillvalen nedan blir utvärderade i den ordning de skrivs på kommandoraden.

TILLVAL och URVALSVILLKOR

-V Skriver ut versionsnumret av kommandot *find*

-name *file* Sann om filen matchar aktuellt filnamn. Vanligt argument-syntax för shell kan användas om den "citeras" (se upp för kolon (:), frågetecken (?) och stjärna (*)).

-perm *onum* Sann om filens tillståndsflaggor exakt matchar det oktala talet *onum* (se *chmod*). Om onum har ett minustecken som prefix, blir fler bitar signifikanta (0177777) och flaggorna jämförs:
(flags & onum) == onum

-type *x* Sann om filtypen är *x*, där *x* kan vara **b** för blockad specialfil, **c** för teckenvis specialfil, **d** för ett bibliotek, **p** för en pipe (FIFO) fil och **f** för en vanlig fil.

-links (-+)*n* Sann om filen har mindre än, mer än eller lika med *n* länkar.

-user *name* Sann om filen tillhör användaren *name*. Om *name* är numeriskt och inte finns som login-namn i filen **/etc/passwd**, antas det vara ett användar-ID.

-group *gname* Sann om filen tillhör gruppen *gname*. Om *gname* är numeriskt och inte finns i filen **/etc/group**, antas det vara ett grupp-ID.

-size (-+)*n(c)* Sann om filen är mindre än, större än eller lika med *n* block lång (512 bytes per block). Om numret direkt åtföljs av tecknet *c* ges storleken i bytes.

-atime (-+)*n* Sann om filen har använts de senaste *n* dagarna, inte använts de senaste *n* dagarna eller användes sist exakt för *n* dagar sedan. Observera att kommandot *find* kommer att ändra användningstiden (access time) till bibliotek när de genomsöks.

-mtime (-+)*n* Sann om filen ändrats de senaste *n* dagarna, inte modifierats de senaste *n* dagarna eller modifierats för exakt *n* dagar sedan.

-ctime (-+)*n* Sann om i-noden för filen ändrats de senaste de senaste *n* dagarna, inte ändrats de senaste **n** dagarna eller ändrats för exakt *n* dagar sedan.

-exec *cmd* Sann om det exekverade kommandot *cmd* ger ett värde noll som utgångsstatus. Slutet på *cmd* måste förses med citerat semikolon (\;). Kommandoargument {} ersätts av aktuellt pathname.

-ok *cmd* Som **-exec** men den genererade kommandoraden visas med ett frågetecken först, och exekveras bara om användaren svarar med att skriva **y**.

-print Alltid sann; skriver ut aktuellt pathname, dvs fullständig sökväg och filnamn.

-cpio *enhet* Alltid sann; skriver aktuell *file* till enheten i *cpio* format (5120 byte per block). Oftast används då tillvalet **-depth** på nästa sida.

-newer *file* Sann om den aktuella filen har modifierats senare än argumentet *file*.

-depth Alltid sann; gör så att genomgången av bibliotekshierarkin görs så att först sker sökningen på innehållet i biblioteket innan biblioteket självt kontrolleras. Detta kan vara användbart när *find* används tillsammans med *cpio* för att flytta filer som finns i bibliotek utan skrivtillstånd. Detta gör det möjligt för filerna att återladdas vid ett senare *cpio* **-i** kommando och lagras i biblioteket innan detta blir skrivskyddat, vilket sker när läsning av biblioteket sker från arkivet.

-mount Alltid sann; Begränsar sökningen till filsystemet som innehåller det specificerade biblioteket eller om inget bibliotek specificerats till nuvarande bibliotek.

-local Sann om filen finns fysiskt på det lokala systemet

\(uttryck \) Sann om uttrycket inom paranteser är sant (paranteser är speciella för shell och måste citeras, dvs föregås av \.

Ovanstående uttryck kan kombineras genom att använda följande operatorer (ordnade i fallande ordning):

negation (!) Att negera ett uttryck görs med utropstecknet (!) som ICKE-operator.

och OCH-operationen åstadkommes genom att placera uttrycken efter varandra på kommandoraden.

eller ELLER-operationen åstadkommes genom att placera operatorn **-o** mellan uttrycken.

FILER

/etc/passwd
/etc/group

EXEMPEL

Exempel 1:

```
find /usr \( -name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
```

Alla filer i biblioteket **/usr** med namnen **a.out**, eller som slutar med ***.o**, och som inte använts den senaste veckan tas bort.

HÄNVISNING

cpio(1), *sh(1)*, *test(1)*, *chmod(1)*

FELMEDDELANDEN

Inga

FIND(1)

NAMN

format - formatera disketter.

SYNTAX

/etc/format [-V] [-tillval] [enhet]

FUNKTION

/etc/format kommandot formaterar disketter med det valda diskettformatet. En diskett måste vara formaterad innan något kan lagras på den. Efter formateringen kan kommandot *tar* användas för att lagra backupfiler på disketten eller kan kommandot *mkfs* användas för att skapa ett D-NIX filsystem.

Om en enhet anges används standard parametrarna, eventuellt ändrade med tillvalsbokstäver, för formateringen.

Om ingen enhet anges, kommer programmet att interaktivt fråga efter enhetsnamnet och formaterings-parametrarna. Se exempel 1.

Standardparametrar för 720 kbytes 5 1/4 tums disketter är:

```
Antal huvuden:          2
Antal cylindrar:        80
Antal sektorer/spår:    9
Sektor storlek:         512
Interleave faktor:  1
```

För att formatera en 5 1/4 tums diskett (1.2 Mbytes) måste antal sektorer/spår ändras till 15, antingen med tillvalet **-r** eller interaktivt. Använd parametrarna nedan.

Parametrar för 1.2 Mbytes 5 1/4 tums disketter är:

```
Antal huvuden:          2
Antal cylindrar:        80
Antal sektorer/spår:    15
Sektor storlek:         512
Interleave faktor:  1
```

För en 640 kbytes diskett, används följande parametrar. Gå ifrån standard med tillvalet **-r** och **-s** eller interaktivt.

Parametrar för 640 kbytes 5 1/4 tums disketter är:

```
Antal huvuden:          2
Antal cylindrar:        80
Antal sektorer/spår:    16
Sektorstorlek:          256
Interleave faktor:  1
```

Parametrar för 3 1/2-tums disketter är:

	720 kbytes	1.44 Mbytes
Antal huvuden:	2	2
Antal cylindrar:	80	80
Antal sektorer/spår:	9	18
Sektorstorlek:	512	
Interleave-faktor:	1	1

TILLVAL

-V Visar versionsnumret på kommandot *format*

-v Visar formateringsparametrarna och används endast i kommandomod. I interaktivt mod visas de alltid.

-h *n* Sätter antal huvuden (sidor) till *n*.

-c *n* Sätter antal cylindrar till *n*.

-r *n* Sätter antal sektorer/spår (records) till *n*.

-s *n* Sätter sektorstorleken till *n* (endast 128, 256, 512 eller 1024).

-i *n* Sätter interleave-faktorn till *n*. Skall för bästa prestanda vara 1.

FILER

/dev/mf*n* *n*=0,1,... diskett-enhet

EXEMPEL

Exempel 1:

Formatera en 720 kbytes 5 1/4 tums diskett i interaktivt mod. Kommandot ges utan parametrar och dialogen blir som nedan, där enheten blir given och accepterad och parametrarna godtagna, innan formateringen startar.

```
/etc/format
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 disc ......
Disk formatted
```

Exempel 2:

För att formatera en diskett med standardparametrar, gör som i exempel 1 ovan, men i direkt mod.

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

Exempel 3:

För att formatera en 1.2 Mbytes diskett:

```
/etc/format -r 15 /dev/mf0
```

Exempel 4:

För att formatera en 640 kbytes diskett med 16 sektorer/spår och 256 bytes/sektor (SBC-DOS format):

```
/etc/format -r 16 -s 256 /dev/mf0
```

HÄNVISNING

mkfs(1M), tar(1)

FELMEDDELANDEN

```
Format error n, cylinder c, side s
```

där de givna värdena är:

n D-NIX error code
c cylinder nummer
s sidnummer

ANMÄRKNING

Detta kommando är systemberoende.

NAMN

fsck - kontroll av filsystem.

SYNTAX

/etc/fsck [-V -y -n -s -t -rr] filsystem ...

FUNKTION

Kommandot *fsck* granskar de angivna filsystemen och rättar interaktivt eventuella fel. *fsck* ignorerar villkorsflaggan 'filsystemet är rent'. Denna flagga sätts om kommandot *fsck* avslutas på ett korrekt sätt.

Filsystemet får inte vara aktiverat (mountad) när det kontrolleras av *fsck*. Vilket annat system som helst utom rootfilsystemet kan kontrolleras från operativsystemnivån.

Parametern filsystem skall ha endera av två utformningar. Om *fsck* exekveras från operativsystemet, på en eller fleranvändarnivå, skall enhetsnamnen anges, t.ex. **/dev/mf0**. Om inget filsystem anges i kommandot *fsck*, kommer *fsck* att läsa den lista över filsystem som finns i filen **/etc/checklist**, ett namn per rad.

Observera: Om root-filsystemet skall kontrolleras måste *fsck* exekveras från laddningsprogrammet. Gör i så fall en manuell uppstart innan *fsck* startas och använd alltid tillvalet **-rr**. Om *fsck* utförs från laddningsprogrammet, skall namnen på filsystemen anges i formatet som används vid manuell systemstart för det aktuella systemet. Namnet kan variera mellan olika system. T.ex. si(16,0), si(2,0), si(0,0), mf(8,0) eller mf(64,0). Endast super-user kan exekvera *fsck*.

För de filsystem som är korrekta kommer en utskrift att göras på antalet filer som ingår, antalet block som används och antalet fria block. Ett block innehåller alltid 512 bytes. Innan en ändring görs i ett filsystem som inte är korrekt, måste operatören oftast bekräfta denna ändring. Data går förlorad vid de flesta ändringar, men detta rapporteras. Tillvalet **-n** blir automatiskt satt av *fsck* vid kontroll av ett filsystem där ingen skrivning är tillåten.

Funktionen hos *fsck* beskrivs mer i detalj i **Systemadministration**.

Filsystemen kontrolleras efter följande lista:

1. Volymbeskrivningsblockets integritet och blockstorlek.
2. Block som mer än en inod gör anspråk på.
3. Block som en inod utanför filsystemets område gör anspråk på.
4. Felaktigt antal länkar.
5. Storleks-kontroll:

   Felaktigt antal block i en fil.

   Bibliotekstorleken är inte en multipel av 16 bytes.
6. Felaktigt format på en inod.
7. Block som inte har någon tillhörighet.

8. Bibliotekskontroll:

   Filnamn som pekar på en inod som inte är allokerad.

   Inod-numret ligger inte inom gränserna.

9. Volymkontroll:

   Mer än 65536 stycken inoder.

   Det finns fler inodsblock än det ingår i filsystemet.

10. Felaktig bitmap.

11. Beräkning av totala antalet fria block och/eller antalet fria inoder saknar överensstämmelse.

Nya filer och bibliotek som är reserverade men inte har någon referens, kan med operatörens tillåtelse, åter läggas till filsystemet genom att de läggs i biblioteket **lost+found**. Det namn som används i detta fall är inodnumret. Enda begränsningen är att biblioteket **lost+found** skall vara placerat i rootbiblioteket i det filsystem som kontrolleras.

TILLVAL

-V Skriver versionsnumret för kommandot *fsck*

-y Antar att alla frågor besvaras med ja.

-n Antar att alla frågor besvaras med nej.

-s Ignorerar nuvarande bitmap och konstruerar en ny. Filsystemet som undersöks bör inte finnas inom systemet när detta tillval är satt eller också måste filsystemet "gömmas" på någon plats i systemet. Direkt efter att kommandot är utfört måste systemet startas igen. Denna procedur är till för att ingen gammal, dålig kopia av bitmappen skall fortsätta att användas eller skrivas till det nya filsystemet.

-t file Detta tillval anger att nästa-argument (file) skall användas som namn på en arbetsfil. Filen används om kommandot *fsck* inte har tillräckligt minnesutrymme för sina tabeller. Filen bör inte befinna sig på det filsystem som undersöks. När kommandot *fsck* är fullbordat kommer filen att tas bort om det är en specialfil eller om filen tidigare inte existerade. Om inte tillvalet **-t** är angiven och kommandot *fsck* behöver en arbetsfil, kommer den att begära att få en från operatören.

-rr Får endast användas under laddningsprogrammet för reparationer av rootfilsystemet. Med **-rr**, blir inga frågor ställda.

FILER

/etc/checklist standardlista över de filsystem som skall kontrolleras. Ett filsystemnamn per rad i filen.
lost+found Bibliotek för återskapade filer.

EXEMPEL

Exempel 1:

Kontrollera ett filsystem på en diskett.

Disketten får inte vara tillgänglig (mounted). Vi antar att disketten är insatt i ett 5 1/4 tums diskettfack. Starta med:

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

Exempel 2:

Kontrollera filsystemet på en intern winchesterskiva.

I detta fall måste systemet stängas av och återstartas i manuell mod till bootnivå 1 (laddningsprogrammet) innan kommandot *fsck* kan användas.

HÄNVISNING

badblk(1M), *fscl(1M)*, *mount(1M)*, *umount(1M)*.

FELMEDDELANDEN

fsck lämnar felmeddelande för alla typer av fel.

ANMÄRKNING

fsck är systemberoende.

FSCK(1)

NAMN

fscl (file-system clean) - testar om filsystemet på en disk är rent.

SYNTAX

/etc/fscl [-V] enhet

FUNKTION

Kommandot *fscl* kontrollerar om flaggan 'file-system-clean' är satt på enheten. Filsystemet på den angivna enheten förutsättes vara rent om flaggan är satt. Super-user privilegier krävs för att använda *fscl*.

Kommandot returnerar utgångsstatus med följande betydelse:

0 = Flaggan 'file-system-clean' är satt.

1 = Flaggan 'file system-clean' är inte satt.

2 = *fscl* kan inte öppna och/eller läsa enheten.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *fscl*

FILER

Inga

EXEMPEL

Exempel 1:

```
/etc/fscl /dev/mf0
```

Om enheten /dev/mf0 är mountad och filsystemet är OK erhålls följande:

```
fscl: File system on device /dev/mf0 is ok!
```

Om filsystemet inte är OK erhålls:

```
fscl: File system on device /dev/mf0 is not ok!
```

Om enheten inte är mountad erhålls följande:

```
SA:drive offline
fscl: Error at read
```

Exempel 2:

```
/etc/fscl /dev/si100
```

/dev/si100 är en enhet som inte är inlänkad i systemet, då erhålls följande felmeddelande:

```
fscl: cannot open device /dev/si100.
```

HÄNVISNING

fsck(1M), mntchk(1M)

FELMEDDELANDEN

Felmeddelande erhålls om enheten inte är mountad i systemet.

ANMÄRKNING

Detta kommando är systemberoende.

NAMN

fsize - Skapar/ta bort mjukvarustyrd checksumma i filsystemet.

SYNTAX

fsize [-V] enhet [nystorlek | max | chk]

FUNKTION

Kommandot *fsize* skapar och kopplar på (eller kopplar bort) mjukvarustyrd checksummeberäkning i ett filsystem på enheten enhet. Samtidigt kan enhetens fysiska storlek ändras genom att en ny storlek anges nystorlek. Storleken motsvarar den som rapporteras av kommandona *mkfs*, *fsck* eller *df*. Den andra parametern kan vara en av följande:

- nystorlek anges normalt i block, men kan istället följas av suffixet k för kbytes eller M för Mbytes. Notera att om nystorlek är mindre än tidigare storlek så kommer eventuella sektorer i tidigare filer som ligger utanför den nya storleken att försvinna, vilket rapporteras av *fsck*, som alltid bör köras efter *fsize*.

- Ordet max kan anges istället för en ny storlek, varvid *fsize* söker enhetens fysiska storlek och sätter filsystemets storlek enligt denna. Samtidigt kopplas checksummeberäkningen bort.

- Ordet chk istället för en ny storlek kopplar på checksummeberäkningen och reserverar ett antal sektorer i slutet för checksummorna, varvid filsystemets storlek minskar något. Check-summesektorerna initieras därefter, vilket tar lite tid.

Om den andra parametern saknas visas nuvarande storlek på filsystemet samt om checksummeberäkningen är inkopplad (on) eller ej (off). Utan parametrar visas en kort beskrivande text.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *fsize*

HÄNVISNING

mkfs(1M), df(1M), fsck(1M)

ANMÄRKNING

Detta kommando är systemberoende.

FSIZE(1)

NAMN

getopt - avkodar kommandooptioner i shellprocedurer.

SYNTAX

set -- ègetopt [-V] tillvalssträng $*è

FUNKTION

getopt används för att bryta ner kommandorader för enkel avkodning i en shell-procedur och samtidigt kontrollera giltigheten i tillvalen. *getopt* ger en sträng av positionsparametrar skapade från de ursprungliga till *getopt*. Med kommandot set i en shell-procedur som i syntaxen ovan, omdefinieras positionsparametrar för senare användning.

'Tillvalssträng' är en sträng av aktuella tillvalsbokstäver. Om en bokstav följs av ett kolon förutsätts tillvalet ifråga följas av ett argument med eller utan föregående mellanslag (space). Det speciella tillvalet -- används för att markera slutet på tillvalen. Om inte angiven i ursprungliga argumenten, genereras den av *getopt*.

Positionsparametrarna ($1 $2 ...) i shell omformas så att varje tillval föregås av ett bindestreck (-) och är i sin egen positionsparameter; varje tillvalsargument placeras i en egen positionsparameter. Specialtillvalet -- ges alltid vid slutet av tillvalen, före de normala positionsparametrarna från den ursprungliga kommandosträngen.

TILLVAL

-V	Skriver versionsnumret av kommandot *getopt*. **-V** måste då ges som första argument till *getopt*.

EXEMPEL

Följande kodfragment visar hur man kan processa argument till ett kommando som kan ha tillvalen **a** och **b**, liksom även tillvalet **o**, vilket måste ha ett argument.

```
set -- ègetopt ao:b $*è
if [ $? != 0 ]
then
        echo $USAGE
        exit 2
fi
for i in $*
do
        case $i in
        -a | -b)    FLAG=$i ; shift ;;
        -o)         OARG=$2 ; shift ; shift ;;
        --)         shift ; break ;;
        esac
done
```

Koden accepterar ett av följande alternativ som likvärdiga:

```
cmd -aoarg file file
cmd -a -o arg file file
cmd -oarg -a file file
cmd -a -oarg -- file file
```

HÄNVISNING

sh(1), set(1)

FELMEDDELANDEN

getopt skriver ut felmeddelande på standard error output när den påträffar en tillvals-bokstav som inte finns i tillvalssträng.

NAMN

getopts - avkodar kommandotillval.

SYNTAX

getopts [-V] tillvalssträng namn [arg ...]

FUNKTION

getopts kan användas i shell-procedurer för att avkoda positionsparametrar och tillåtna tillvalssträngar. Det stöder alla regler i standarden för kommandosyntaxen. Det bör användas istället för kommandot *getopt*.

tillvalssträng måste innehålla alla de tillval som är tillåtna. Om ett tillval följs av ett kolon, så förväntas optionen ha ett argument eller en grupp argument som då måste vara separerade från tillvalstecknet med tab eller mellanslag.

Varje gång getopts anropas placeras nästa tillval i shell-variabeln name och indexet till nästa parameter som skall processas placeras i shell-variabeln OPTIND. Varje gång shell eller en shell-procedur startas, initieras OPTIND till 1.

När ett tillval behöver ett tillvals-argument placerar *getopts* detta i shell-variabeln OPTARG.

Om man stöter på ett ogiltigt tillval placeras ett ? i name.

När alla tillval är slut avslutas *getopts* med en utgångsstatus skild från noll. Specialtillvalet "--" kan användas för att ange slut på tillvalen.

getopts avkodar normalt positionsparametrarna. Om extra parametrar ges på *getopts* kommandorad avkodas dessa istället.

Alla nya kommandon skall följa standarden för kommandosyntax och de bör använda *getopt* eller *getopts* för att avkoda positionsparametrar och kontrollera vilka tillval som är tillåtna för det kommandot.

TILLVAL

-V	Skriver versionsnumret av kommandot *getopts*. **-V** måste då ges som första argument till *getopts*.

EXEMPEL

Följande kodfragment visar hur man kan processa parametrar till ett kommando som kan ha tillvalen **a** och **b**, liksom även tillvalet **o**, vilket måste ha ett argument.

```
while getopts abo: c
do
   case $c in
        a | b)      FLAG=$c;;
        o)          OARG=$OPTARG;;
        \?)         echo $USAGE
                    exit 2;;
   esac
```

```
done
shift 'expr $OPTIND - 1'
```

Koden accepterar följande alternativ som likvärdiga:

```
cmd -a -b -o "xxx z yy" fil
cmd -a -b -o "xxx z yy" -- fil
cmd -ab -o xxx,z,yy fil
cmd -ab -o "xxx z yy" fil
cmd -o xxx,z,yy -b -a fil
```

HÄNVISNING

sh(1), set(1), getopt(1)

ANMÄRKNING

Trots att följande lättnader i syntax reglerna finns bör de inte användas då de kanske inte kommer att stödjas i senare versioner av systemet. Som i exemplen ovan är a, b och o tillval, och tillvalet o kräver ett tillvalsargument:

```
cmd -aboxxx file
```

Bryter mot regeln att tillval med tillvalsargument inte får grupperas med andra tillval.

```
cmd -ab -oxxx file
```

Bryter mot regeln att det måste finnas tab eller mellanslag efter ett tillval som behöver ett tillvalsargument.

Om värdet på shell-variabeln OPTIND ändras eller om man avkodar olika parametersträngar kan man få oväntade resultat.

Kommandot *getopts* är internt i shell och exekveras utan att en ny process skapas.

FELMEDDELANDEN

getopts skriver ut felmeddelanden till standard error när den träffar på en tillvals-bokstav som inte finns i tillvalssträng.

NAMN

getty - bestämmer terminaltyp, mod, hastighet, och linjetyp samt startar en login-procedur.

SYNTAX

/etc/getty [-V] [-h] [-m] [-t *timeout*] [-l] [-r] [-w *sec*] [-b] [-B *text*] [-u *period*] enhet [hastighet [typ [*linedisc*]]]

/etc/getty -c file

FUNKTION

Kommandot *getty* är ett program som startas av *init*. Det är den andra processen i serien (*init-getty-login*-shell) som slutligen ger en användare åtkomst till operativsystemet. Först väntar *getty* eventuellt ett antal sekunder givet med tillvalet **-w**, därefter väntar *getty* tills bärvågsindikering (DCD) erhållits från linjen, varefter ett meddelande med systemnamnet genereras. Systemnamnet är en sträng, som ges av systemanropet *uname*. Om filen **/etc/issue** finns, skrivs den därefter ut till användarens terminal följt av login-frågan, erhållet från **/etc/gettydefs**. *getty* läser användarens login-namn och startar kommandot *login* med användarnamnet som argument. Samtidigt som namnet läses in försöker *getty* att anpassa systemet till använd terminaltyp och hastighet. Med tillvalet **-r** väntar *getty* tills något tecken kommit in från linjen innan något meddelande genereras. Med tillvalet **-b** (eller **-B**) väntar *getty* tills ett CR-tecken kommit och ställer in baudrate genom att testa på CR-tecknet. Tillvalet **-m** anger att porten som *getty* startas mot redan är öppen, t.ex. när *getty* startas av *mgetty*. Med tillvalet **-u** anges att en låsfil skall skapas som hindrar att flera försöker använda en linje samtidigt. Denna låsfil är generell och används även av *cu, kermit* och *uucp* m fl.

Enhet är namnet på en terminalport i **/dev** mot vilken *getty* skall arbeta. *getty* använder denna sträng som namn på en enhet i biblioteket **/dev** som skall öppnas för läsning och skrivning. Såvitt inte *getty* är startat med tillvalet **-h**, kommer *getty* att göra ett hangup på linjen genom att sätta hastigheten till 0 innan den sätter standard eller angivet hastighetsvärde. Tillvalet **-t** jämte timeout i sekunder anger att *getty* skall koppla ner linjen om ingen skriver något inom angiven tid.

Argumentet hastighet är en pekare till en tabell i filen **/etc/gettydefs** med definition av hastighet och andra terminalparametrar. Dessa definitioner ger överföringshastighet vid start, hur loginfrågan skall se ut, terminalportens standardparametrar samt vilken nästa hastighet som skall väljas om användaren anger starthastigheten som otillfredsställande genom att ge ett BREAK-tecken. Förutom detta kan filen **/etc/gettydefs** innehålla frågesträngar som *getty* sänder till kommandot *login*, som tillvalen **-L**, **-P** och **-D**, vilket gör att valfritt språk kan användas under inloggningen för login-frågan, frågan efter lösenord och sekundärt lösenord. Se kommandot *login* och **/etc/gettydefs** i Systemadministration.

Argumentet typ är en teckensträng som för *getty* anger typ av terminal ansluten till linjen. *getty* accepterar följande typer:

none	standard
vt61	DEC vt61
vt100	DEC vt100
hp45	Hewlett-Packard HP45
c100	Concept 100

Standard terminal är none, dvs vilken bildskärm eller normal terminal som helst, ej känd för systemet. För att terminaltypen skall ha någon mening måste de virtuella terminalhanterarna vara kompilerade i operativsystemet. För närvarande är de inte inkompilerade.

Argumentet *linedisc* är en teckensträng som beskriver vilken linjedisciplin som skall användas vid kommunikation med terminalen. Även linjedisciplinen hämtas från operativsystemet. För närvarande är dock endast en tillgänglig, LDISC0(nolla), vilken också är standard.

Om inga argument ges sätter *getty* hastigheten till 300 baud, anger användning av raw mod (ett tecken i taget), inget eko, alla pariteter accepteras, newline tecken ändras till return-linefeed (CRLF) och tabtecken expanderas på standard output.

Loginmeddelandet skrivs ut innan användarnamnet läses tecken för tecken. Om ett null-tecken (eller 'frame'-fel) erhålles tyder detta på att användaren tryckt BREAK. Detta får *getty* att försöka med nästa hastighet i serien. Sekvensen som *getty* försöker med bestäms av vad som finns angivet i **/etc/gettydefs**.

Användarnamnet avslutas med ett tecken för new-line eller carriage return. Det senare får systemet att fortsättningsvis behandla returntecknet korrekt.

Om inte tillvalet **-l** angivits, genomsöks användarnamnet för att se om det innehåller några gemena (lower-case) tecken. Om inte och om namnfältet inte är tomt transformerar systemet inmatade versala tecken till motsvarande gemena i fortsättningen. Dessutom transformeras alla utmatade tecken till versaler.

Under inmatning av användarnamn och lösenord används tecknen **#** och **@** för erase och kill om dessa tecken inte omdefinierats i filen **/etc/gettydefs**.

Slutligen anropas *login* av *getty* med användarnamnet som argument. Ytterligare argument kan skrivas efter login-namnet. Dessa överförs till *login*, som placerar dem i miljön (se *login*).

Ett test-tillval finns även. När *getty* ges med tillvalet **-c** och ett filnamn, genomsöks filen som om denna var **/etc/gettydefs** och resultatet skrivs ut på standard output. Om okända moder eller felaktiga angivelser upptäcks meddelas detta. Om allt är korrekt skrivs värdet för olika flaggor ut. Se kommandot *stty* eller *ioctl*(2) för värdenas betydelse. Observera att vissa värden blir automatiskt tillagda.

TILLVAL

-V Skriver ut versionsnumret för kommandot *getty*

-c *filnamn* Genomsöker angiven fil och kontrollerar att den har rätt format för att användas som filen **/etc/gettydefs**.

-h Hindrar hangup på linjen innan loginproceduren startas.

-t *sec* Specificerar en tidsgräns i sekunder, efter vilken tid linjen kopplas ned om ingen login sker efter att förbindelsen har etablerats.

-l Hindrar *getty* från att automatiskt transformera mellan versaler och gemena, om login-namnet avgivits med enbart versaler.

-r Hindrar *getty* från att skriva ut något meddelande, förrän minst ett tecken lästs från linjen. Detta tillval kan användas om två datorer kopplas ihop direkt via två terminalportar, dvs då två *getty*-processer står mot varanda.

-w *sec* *getty* väntar *sec* sekunder efter start innan den börjar arbeta.

-b *getty* känner av baudrate på porten genom att vänta tills användaren skriver CR (carriage return) på terminalen. I motsvarande tabell i **/etc/gettydefs** med terminalparametrar får då inte ges några hastighetsparametrar.

-B *text* Som tillvalet **-b**, men *getty* sänder först textsträngen *text* till terminalen för att göra användaren uppmärksam på att datorn är beredd att ta emot ett CR. Denna sträng sänds med en standardbaudrate och kan se konstig ut på terminaler med annan baudrate.

-u *period* Anger att *getty* skall vänta om någon annan använder linjen och att en låsfil skall skapas då förbindelsen etableras, dvs då bärvågsindikering (DCD) erhålles. Även kommandona *cu*, *kermit* och *uucp* m fl använder samma låsfil. Om linjen är upptagen, dvs en giltig låsfil finns, väntar *getty* och testar låsfilen regelbundet tills den blir ogiltig eller tas bort. Argumentet *period* anger periodtiden i sekunder, normalt 60 sekunder. En låsfil innehåller ägarens process-id och är giltig enbart om ägarprocessen är aktiv i systemet. Lås-filens namn bildas enligt terminalportens namn: **/usr/spol/locks/LCK..ttyXX**.

Då tillvalet **-u** ges är det viktigt att inkopplade modem byglas för att inte ge DCD aktivt innan uppringningssekvens eller inkommande anrop sker, för att inte linjen ska låsas.

FILER

/etc/gettydefs
/etc/issue
/etc/inittab

HÄNVISNING

init(1M), login(1), tty(1), uname(1)

ANMÄRKNING

Automatisk avkänning av CTRL-H/CTRL-U samt ESS2-protokollet är inte implementerat. Istället kan dessa kontrolltecken anges direkt i filen **/etc/gettydefs**.

För att mottagning av inkommande samtal och utringning ska kunna använda samma port, får eventuellt inkopplat modem inte ligga med DCD hög i väntläget, utan enbart då modemet är aktivt i dataläge eller vid nummerslagning.

Tillvalen **-L**, **-M**, **-X** och **-R**, som också finns, används enbart tillsammans med *dcall*, se *dcall* för mer information.

NAMN

grep - söker efter mönster i filer.

SYNTAX

grep [-Vblcnsvi] uttryck [files]

FUNKTION

Kommandon inom familjen *grep* undersöker indata från filer (standard input) efter rader som passar ett mönster. Normalt kopieras varje funnen rad till standard output. grep-mönster är begränsade reguljära uttryck liknande de som används i *ed*. Jämför kommandot *fgrep*.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *grep*.
-v	Alla rader utom de som matchar skrivs ut.
-c	Endast antalet matchande rader skrivs ut.
-i	Läser versaler och gemener som lika vid sökningen.
-l	Endast namnen på de filer, som innehåller matchande rader, skrivs ut (en gång) separerade med radslutstecken.
-n	Varje rad föregås av sitt relativa radnummer i filen.
-b	Varje rad föregås av numret på det block där raden hittades. Detta är ibland användbart när man vill söka ett block med visst innehåll på ett skivminne.
-s	Felmeddelande som normalt ges för icke-existerande eller oläsbara filer visas ej.

Filnamnet skrivs alltid ut om det finns mer än en indata fil. Man bör vara försiktig vid användandet av tecknen **$**, *, **[**, **^**, **|**, **(**, **)** och \ eftersom de också har betydelse för shellprogram. Man bör därför omge hela uttrycket med apostrofer '...'.

Ordningsföljden för prioritering av oparatorer är **[]**, sedan ***?+**, därefter sammanfogningar och sist **|** och new-line.

HÄNVISNING

ed(1), fgrep(1), sed(1), sh(1)

FELMEDDELANDEN

Utgångsstatus är 0 om någon rad matchar, 1 om ingen matchar, 2 för syntax-fel eller oåtkomliga filer (även om rader hittats i andra filer).

ANMÄRKNING

Radlängden begränsas till BUFSIZ tecken; längre rader trunkeras. (BUFSIZ är definierad i **/usr/include/stdio.h.** och är normalt 512 tecken).

Om det finns en rad med inbäddade nollor, kommer *grep* bara att söka upp till första nollan; om den matchar skrivs hela raden ut.

NAMN

init, telinit - initierar systemprocesser.

SYNTAX

/etc/init [-V] [0123456sqhabc]

telinit [-V] [0123456sqhabc)

FUNKTION

init är en allmän processgenerator. Dess främsta uppgift är att skapa processer från ett script som finns i filen **/etc/inittab**. I denna fil anges bl a att *init* skall starta *getty's* på varje linje som en användare kan logga in på. *init* kontrollerar också de självstyrande automatiska processer som behövs i ett visst system.

init förutsätter vid varje tillfälle att systemet är i en viss systemnivå (runlevel). Systemnivåer kan betraktas som konfigurationer av program i systemet, där varje konfiguration endast tillåter en bestämd grupp av processer att vara aktiva.

De processer som *init* skapar för var och en av dessa systemnivåer definieras i filen **inittab**. *init* kan vara i en av åtta systemnivåer, 0-6, s och h . Systemnivåer ändras genom att en användare med behörighet startar programmet **/etc/init** (vilket är länkat till **/etc/telinit**). Detta användarskapade *init* skickar signaler till det ursprungliga *init*, startat av operativsystemet vid senaste systemstart. Därmed ändras till en ny systemnivå.

init startas av operativsystemet som sista steg i boot-processen. Det första *init* gör är att söka efter **/etc/inittab** och se om där finns angivet någon post som märkts *initdefault.* Om så är fallet använder *init* den systemnivå som specificerats där som systemnivå. Om inget *initdefault* är angivet i **inittab** eller om **inittab** saknas, begär *init* att användaren skall ange en systemnivå från den virtuella systemkonsolen, **/dev/syscon**. Om ett **s** ges går *init* endast in i enanvändarnivån (single-user level). Detta är den enda systemnivå som inte kräver att filen **inittab** existerar. Om filen **/etc/inittab** inte finns är därför enanvändarnivån den enda tillåtna systemnivån som *init* kan gå in i. I enanvändarnivån öppnas den virtuella konsolterminalen **/dev/syscon**, för läsning och skrivning och kommandot **/bin/su** exekveras omedelbart. Enanvändarnivån kan avslutas på två sätt: Om shell termineras med ett end-of-file, kommer *init* att begära en ny systemnivå. Alternativt kan kommandot *init* eller *telinit* ges för att framtvinga en ändring av systemnivån.

Om ingen fråga kommer från *init* om ny systemnivå när systemet bootas utan en giltig **/etc/inittab**-fil, kan detta bero på, att **/dev/syscon** är länkat till en annan enhet än den fysiska huvudkonsolen (**/dev/systty** vilken ska vara samma som **/dev/console**). I detta fall kan *init* tvingas att länka om **/dev/syscon** genom att operatören trycker på tangenten DEL på huvudkonsolen.

När *init* frågar efter den nya systemnivån kan operatören ange endast en av siffrorna 0-6 eller någon av bokstäverna h eller s. Om s ges kommer *init*

att arbeta i enanvändarmod som beskrivits ovan, men med resultatet att **/dev/syscon** blir länkat till användarens terminallinje, vilken alltså definieras om som ny virtuell systemkonsol. Ett meddelande ges på den fysiska huvudkonsolen, **/dev/systty, (=/dev/console)** som säger var den virtuella terminalen nu är lokaliserad. Om h anges stoppas systemet.

Som terminalparametrar för den virtuella systemkonsolen (**/dev/syscon**) används, när *init* startas och när systemnivån ändras till enanvändarmod, de parametrar som angivits i andra fältet i tabellen för console i filen **/etc/gettydefs**. Om denna fil eller denna tabell inte finns används standardvärden (9600 baud).

Om 0 till 6 anges som svar går *init* in i motsvarande systemnivå. Några andra indata godkännes inte, varvid frågan skrivs ut igen. Om det är första gången som *init* går in i en systemnivå annan än enanvändarmod(s), kommer *init* att först söka i **/etc/inittab** för speciella poster av typen boot och bootwait. Dessa utföres under förutsättning att angiven systemnivå motsvarar vad som angivits innan någon normal genomgång av **inittab** sker. På detta sätt kan speciell initiering av operativsystemet ske (exempelvis mountning av filsystem), innan användare får tillgång till systemet. **Inittab** genomsöks för alla processer som skall startas på systemnivån ifråga.

Systemnivå 2 är normalt definierad som fleranvändarnivå och innehåller alla de terminalprocesser och residenta bakgrundsprocesser som ska vara aktiva i fleranvändarmiljö.

I fleranvändarmiljö anger filen **inittab** vanligen att *init* ska skapa en login-process för varje terminal i systemet.

I terminalprocesser kommer shell att terminera (utloggning) pga ett end-of file, antingen skrivet på tangentbordet eller genererat av ett hangup på terminallinjen. När *init* får en signal, som säger att processen den skapade terminerades, skrivs detta och orsaken därtill i filerna **/etc/utmp** och **/etc/wtmp** om dessa finns (see *who*). En loggbok över skapade processer bevaras i filen **/etc/wtmp** om denna finns.

För att skapa var och en av processerna i filen **inittab** läser *init* varje post. För varje post som specificerar en process som skall startas, skapas en ny process med systemanropet fork. Sedan den har skapat alla processer specificerade i **inittab** avvaktar *init* signaler om att någon av underprocesserna avslutats, en powerfail-signal eller en signal från *init* eller *telinit* att byta systemnivå. När något av dessa förhållanden är för handen går *init* åter igenom **inittab** för kontroll. Nya angivelser kan när som helst läggas till **inittab**. *init* avvaktar dock att en av de tre signalerna ovan ges. För att aktivera *init*, kan kommandot *telinit* **q** användas. Det får *init* att omedelbart kontrollera filen **inittab**.

Om *init* erhåller en powerfail signal (SIGPWR) och inte är i enanvändarmod söker det igenom **inittab** för att hitta särskilda poster som skall aktiveras vid powerfail. Dessa startas (om systemnivån så tillåter) innan något annat utförs. På detta sätt kan *init* utföra olika rensnings- och uppdateringsprocesser om operativsystemet får en signal om avbrott i spänningsförsörjningen.

Om *init* får en begäran att ändra systemnivå (via *telinit*), sänder *init* en varningssignal (SIGTERM) till alla processer som inte är definierade på begärd nivå. *init* väntar 10 sekunder innan dessa processer tvingas stoppa med kill-signalen (SIGKILL).

telinit, är länkat till **/etc/init** för att styra *init's* aktiviteter. Det avkodar argumentet (ett tecken) och signalerar *init* via systemanropet *kill* att utföra avsedd aktivitet. Följande argument är direktiv till *init (telinit)*.

0-6	Beordrar *init* att ändra systemnivån till en av nivåerna 0-6.
a,b,c	Beordrar *init* att exekvera endast de poster i filen **/etc/inittab**, som har systemnivåerna **a,b** eller **c** satta.
q	Beordrar *init* att åter gå igenom filen **/etc/inittab**.
s	Beordrar *init* att ändra till enanvändarnivå. När denna nivå har antagits ändras virtuella huvudkonsolen **/dev/syscon**, till den terminal från vilken kommandot har getts.
h	Stänger omedelbart av systemet och stoppar processorn. Filsystemet stängs av korrekt. **OBS!** Använd detta kommando bara vid fel. Normalt stoppas systemet med nyckeln eller med **/etc/shutdown -k**.

telinit kan endast utföras av en super-user eller en medlem av gruppen sys.

Kommandot *who* **-r** ger information om den aktuella systemnivån.

TILLVAL

-V	Skriver versionsnumret på kommandot *init* eller *telinit*

FILER

/etc/inittab
/etc/utmp
/etc/wtmp
/dev/syscon
/dev/systty = /dev/console

HÄNVISNING

getty(1M), login(1), sh(1), who(1), kill(1)

FELMEDDELANDEN

Om *init* finner att det håller på att upprepade gånger omstarta en process enligt **/etc/inittab** mer än 10 gånger i följd inom 2 minuter, tyder detta på en felaktighet i kommandosträngen. Det genererar då ett felmeddelande på huvudkonsolen och vägrar utföra ny omstart tills dess att antingen 5 minuter har gått eller signal erhålles från operatören (*telinit*). Detta hindrar *init* från att belasta systemets resurser om någon har gjort en felskrivning i filen **/etc/inittab** eller om ett program som **inittab** refererar till har tagits bort.

INIT(1M)

NAMN

ipcrm - ta bort en meddelandekö, en semaforgrupp eller id för ett delat minnessegment.

SYNTAX

ipcrm [-V] [-m *shm_id*] [-q *msg_id*] [-s *sem_id*] [-M *shm_key*] [-Q *msg_key*] [-S *sem_key*] ....

FUNKTION

ipcrm tar bort identifierare för en eller flera angivna meddelanden, semaforer eller för delade minnessegment. Identifierarna specificeras med följande tillval.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *ipcrm*.
-q *msg_id*	ta bort meddelandekö-identifieraren *msg_id* från systemet och ta bort meddelandekön och data-strukturen associerade med den.
-m *shm_id*	ta bort identifieraren *shm_id* för det delade minnessegmentet från systemet. Det delade minnessegmentet och dess datastruktur tas bort efter att sista processen kopplats bort (detach).
-s *sem_id*	ta bort identifieraren för semaforen *sem_id* från systemet och ta bort semaforgruppen och datastrukturen associerad med den.
-Q *msg_key*	ta bort meddelandekö-identifieraren, som skapats med nyckeln *msg_key*, från systemet och ta bort meddelandekön och datastrukturen associerad med den.
-M *shm_key*	ta bort identifieraren för det delade minnes-segmentet, som skapats med nyckeln *shm_key*, från systemet. Det delade minnessegmentet och dess datastruktur tas bort efter att sista processen kopplats bort (detach).
-S *sem_key*	ta bort semaforidentifieraren, som skapats med nyckeln *sem_key*, från systemet och ta bort semaforgruppen och datastrukturen associerade med den.

Detaljer om borttagandet beskrivs tillsammans med *msgctl*(2), *shmctl*(2) och *semctl*(2). Identifierarna och nycklarna listas med *ipcs*.

HÄNVISNING

ipcs(1), msgctl(2), msgget(2), msgop(2), semctl(2),semget(2), semop(2), shmctl(2), shmget(2), shmop(2)

IPCRM(1)

NAMN

ipcs - rapportera status för kommunikationen mellan processer i systemet.

SYNTAX

ipcs [-V] [optioner]

FUNKTION

Kommandot *ipcs* skriver ut information om kommunikationsaktiviteter mellan processer i systemet. Utan tillval skrivs informationen i kort format för meddelandeköer, delade minnessegment och aktiva semaforer. Annars styr tillvalen vilken information som skall skrivas ut.

TILLVAL

-V Skriv ut versionsnumret för kommandot *ipcs*

-q Skriv information om aktiva meddelandeköer.

-m Skriv information om aktiva delade minnes-segment.

-s Skriv information om aktiva semaforer.

Om någon av tillvalen **-q**, **-m** eller **-s** anges, kommer enbart information om dessa att skrivas ut. Om inga av dessa anges skrivs information ut om alla dessa tre.

-b Skriv information om största tillåtna storlek. Maximalt antal bytes i meddelanden i meddelandeköer, segmentstorlek för delade minnesegment, och antal semaforer i varje grupp (set) av semaforer. Se nedan vad listans kolumner innehåller.

-c Skriv login-namn och gruppnamn för den som skapade posten. Se nedan.

-o Skriv information om resursanvändningen. Antal meddelanden i kö och totala antal bytes i meddelanden i kö för meddelandeköer och antal processer som delar gemensamma minnessegment.

-p Skriv information om processnumren. Process-ID för senaste process som sände ett meddelande och process-ID för senaste process som tog emot ett meddelande i meddelandeköerna och process-ID för den process som skapade och den som senast anslöt sig eller kopplade bort sig från delade minnessegment. Se nedan.

-t Skriv tidsinformation. Tiden för senast styroperation som ändrade åtkomsttillstånden för alla funktioner. Tiden för senaste msgsnd och senaste msgrcv i meddelandeköerna, senaste shmat och senaste shmdt för delade minnessegmenten, senaste *semop*(2) för semaforer. Se nedan.

-a Använd alla utskriftstillvalen. (Detta är ett kort sätt att ange **-b**, **-c**, **-o**, **-p** och **-t**.)

-C corefile Använd filen **corefile** istället för **/dev/kmem.**

-N namelist Argumentet skall vara ett namn på en alternativ namnlista (**/dnix** är standard).

Kolumntitlarna och informationen i kolumnerna i listningen från *ipcs* ges nedan; texten inom parentes indikerar de tillval som ger motsvarande kolumn i listningen; all betyder att kolumnen alltid listas. Notera att dessa tillval endast bestämmer vilken information som ges för varje funktion; de anger inte vilka funktioner som skall listas.

T	**(all)**	Typ av funktion.
		q meddelandekö
		m delat minnessegment
		s semafor
ID	**(all)**	Identifierare för funktionsposten.
KEY	**(all)**	Nyckeln som används som argument till msgget, semget eller shmget för att skapa funktionsposten. (**Observera:** Nyckeln för ett delat minnessegment ändras temporärt till IPC_PRIVATE då segmenten har tagits bort, tills alla processer, som var kopplade till det, har kopplats bort (detach)).
MODE	**(all)**	Funktionens åtkomsttillstånd och flaggor: Dessa består av 11 tecken som tolkas enligt följande: De första två tecknen är:
	R	om en process väntar med ett msgrcv
	S	om en process väntar med ett msgsnd;
	D	om motsvarande delade minnes-segment har tagits bort. Det kommer att försvinna när sista processen kopplas bort (detach).
	C	om motsvarande delade minnesegment ska nollställas första gången en process kopplar sig till det (attach).
	-	om motsvarande specialflagga inte är satt.
		De följande 9 tecknen tolkas som tre grupper om tre tecken vardera. Första gruppen anger ägarens tillstånd; andra anger tillstånden för andra i användarens grupp för funktionsposterna; och den sista för alla övriga. Inom varje grupp anger första tecknet lästillstånd, andra visar skrivtillstånd eller tillstånd att ändra och tredje tecknet används för närvarande inte.
		Tillstånden indikeras enligt nedan:
	r	för lästillstånd
	w	för skrivtillstånd
	a	för tillstånd att ändra
	-	om motsvarande tillstånd ej finns.

OWNER (all)	Användarnamnet för ägaren till funktionsposten.
GROUP (all)	Gruppnamnet för gruppen som ägaren till funktionsposten tillhör.
CREATOR (a,c)	Användarnamnet för den som skapat funktionsposten.
CGROUP (a,c)	Gruppnamnet för gruppen som den som skapat funktionsposten tillhör.
CBYTES (a,o)	Antalet bytes i meddelanden som just nu finns i meddelandekön.
QNUM (a,o)	Antalet meddelanden som just nu finns i motsvarande meddelandekö.
QBYTES (a,b)	Maximala antal bytes tillåtet i meddelanden i motsvarande meddelandekö.
LSPID (a,p)	Process-ID för senaste process som sänt ett meddelande till motsvarande meddelandekö.
LRPID (a,p)	Process-ID för senaste process som mottagit ett meddelande från motsvarande meddelande-kö.
STIME (a,t)	Tiden då senaste meddelandet sändes till den motsvarande meddelandekön.
RTIME (a,t)	Tiden då senaste meddelandet mottogs från motsvarande meddelandekö.
CTIME (a,t)	Tiden när motsvarande post skapades eller ändrades.
NATTCH (a,o)	Antalet processer som kopplats till motsvarande delade minnessegment.
SEGSZ (a,b)	Storleken av motsvarande delade minnessegment.
CPID (a,p)	Process-ID för den som skapat det delade minnessegmentet.
LPID (a,p)	Process-ID för senaste process som kopplade sig till eller från det delade minnessegmentet.
ATIME (a,t)	Tiden då senaste kopplingen till motsvarande delade minnessegment var klar.
DTIME (a,t)	Tiden så senaste bortkoppling var klar från motsvarande delade minnessegment.
NSEMS (a,b)	Antalet semaforer i gruppen associerad med motsvarande semaforpost.
OTIME (a,t)	Tiden när senaste semaforoperation var klar gentemot gruppen associerad med semaforposten.

FILER

/dnix	systemets namnlista
/dev/kmem	minnet
/etc/passwd	användarnamn
/etc/group	gruppnamn

HÄNVISNING

ipcrm(1), msgop(2), semop(2), shmop(2)

FELMEDDELANDE

Inga

ANMÄRKNING

Situationen ändras medan *ipcs* utförs; bilden som ges är endast en nära approximation av verkligheten.

KERMIT(1)

NAMN

kermit - filöverföring med kermit protokoll.

SYNTAX

kermit [-l enhet] [-b baud] [-p paritet] [-t][-srk] [filnamn] [-a filnamn] [-gfcniwqdh]

FUNKTION

kermit är ett program som används för överföring av filer mellan datorer. Vid överföringen används ett speciellt protokoll för att få felfri överföring. Både programmet och protokollet är utvecklat vid Columbia University. Det här är en version av programmet *kermit*.

kermit kan användas både direkt med kommandoparametrar och interaktivt. För information om den interaktiva användningen samt ytterligare information om programmet, se beskrivningen av *kermit* i **Användarhandboken**.

Vid användning som direktkommando kan man styra både parametrar och händelser man vill ha utförda.

kermit kan användas dels i lokal mod och dels i fjärrstyrd mod. Om inget annat anges startar *kermit* i fjärrstyrd mod.

Observera: Både läs- och skrivtillstånd måste finnas på de terminallinjer som används av *kermit*.

TILLVAL

-s Sänder den fil eller de filer som specificerats. Läser från standard input (ej terminalen) om - anges som filnamn.

-r Tar emot en eller flera filer. *kermit* väntar passivt på filerna.

-k Tar emot en eller flera filer och skickar dem till standard output.

-a Om filöverföring ska ske kan ett alternativt namn på filen anges.

-x Startar 'server' mod. Kan användas både i lokal och fjärrstyrd mod.

-l Specifierar en terminallinje som ska användas för filöverföringen. Exempel:

```
kermit -l /dev/tty05
```

-b Specifierar överföringshastigheten för terminallinjen som valdes med **-l**. Exempel:

```
kermit -l /dev/tty05 -b 9600
```

-p Specificerar pariteten. e, o, m, s, n (even, odd, mark, space, no parity) kan anges. Om pariteten sätts till

något annat än 'no parity' kommer speciella koder (prefix) att användas om 8-bitars data sänds.

-t Specificerar halv duplex överföring.

-g Begär att den fjärrstyrda *kermit* ska sända den specificerade filen. Exempel:

```
kermit -g file.doc
```

-f Sänd stoppkommando (finish) till 'remote server'.

-c Kopplar upp sig för terminalkommunikation innan någon överföring med protokoll sker. Återgå till det lokala systemet med uthopp (normalt CTRL-\, följt av bokstaven c).

-n Som **-c**, fast efter protokollöverföring.

-i Filerna ska sändas exakt som de ser ut. Detta är nödvändigt vid överföring av binära filer.

-w Skrivskydda. Inga befintliga filer skrivs över.

-q Överföring utan att något skrivs ut på skärmen.

-d Debug. Information om vad som sker lagras i filen **debug.log** i det aktuella biblioteket.

-h Hjälp. Visar alla tillgängliga tillval.

FILER

debug.log	
.kermrc	Initieringsfil i användarens hembibliotek.
/usr/spool/locks/LCK..enhet	Låsfil som hindrar dubbel användning av en enhet (ex tty16).

EXEMPEL

Exempel 1:

```
kermit -l /dev/tty05 -b 1200 -cn -r
```

Kopplar upp *kermit* mot den andra datorn via **/dev/tty05** med överföringshastigheten 1200 baud för inloggning och angivande av kommandon. Därefter ges uthopp och filer tas emot. Efter filöverföringen kopplas man upp igen för utloggning och avslutning.

Exempel 2:

```
kermit -l /dev/tty05 -b 1200 -cntp m -r -a foo
```

Som föregående exempel, men överföringen sker med halv duplex och pariteten satt till mark. Den första filen som kommer ska lagras under namnet foo.

Exempel 3:

```
kermit -iwx
```

Startar *kermit* som server. Överföringen sker utan newline/ carriage return-linfeed omvandling. **-i** anges för överföring av binära filer.

Exempel 4:

```
kermit
```

Startar *kermit* i interaktiv mod, varvid *kermit*-kommandon i filen **.kermrc** utförs, om denna fil existerar i användarens hembibliotek.

HÄNVISNING

Kermit-dokumentationen

FELMEDDELANDEN

Inga

NAMN

kill - Avslutar processer

SYNTAX

kill [-V] [-signal] processid ...

FUNKTION

Kommandot *kill* avslutar en process i systemet genom att sända en signal till processen. Kommandot skickar signal nummer 15 som standard till den specificerade processen eller den signal som anges som argument. En del processer ignorerar vissa signaler. Dessa processer kan avslutas med signal -9, vilket är det säkraste sättet att stoppa en process.

Processid är ett nummer som systemet ger en process vid uppstarten. Detta nummer är den enda identifikation som finns till processen. Processid (PID) kan fås fram för alla processer med kommandot **ps.** Om 0 anges som processid i *kill*-kommandot, kommer alla medlemmar i den aktuella processgruppen, dvs alla processer som är aktiverade under nuvarande login, att signaleras.

De processer som skall avslutas måste tillhöra användaren om inte användaren är super-user.

Observera: Användarens egen loginprocess kan avslutas med *kill*, detta innebär att användaren kommer att bli utloggad från systemet. Men denna process kan bara avslutas med signal nummer 9.

Med super-user privilegier kommer kommandot *kill 1* att omedelbart stänga av hela systemet och stoppa processorn, eftersom processid 1 alltid är systemprocessen *init*. Detta förfarande skall emellertid endast användas om något problem på systemet gör det omöjligt att avsluta på ett normalt sätt. Jämför med kommandot *shutdown* och den normala avstängningsproceduren.

TILLVAL

-V	Skriver versionsnumret på kommandot *kill*

FILER

Inga

EXEMPEL

Exempel 1:

```
kill 1076
```

När bakgrundsprocessen startades upp, meddelade systemet att denna process har processid 1076. I och med att detta processid anges, kommer denna bakgrundsprocess att avbrytas och avslutas.

Exempel 2:

```
kill -9 107
```

Om bakgrundsprocessen med numret 107 inte kan avslutas med enbart *kill*, måste signalen -9 skickas till processen för att den skall kunna avslutas.

HÄNVISNING

sh(1), ps(1), shutdown(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

I systemet finns flera olika signaler. De som vanligen används med kommandot *kill* är följande:

-1	Hangup (SIGHUP). Motsvarar signalen från en modemkopplad terminalport då förbindelsen bryts.
-2	Avbrott (INTR). Motsvarar normalt tangenten DEL på tangentbordet (Se *stty*).
-3	Uthopp (QUIT). Motsvarar normalt tangenten CTRL-\ på tangentbordet. (Se *stty*).
-9	Stopp (SIGKILL). Denna signal kan inte fångas (trap) eller ignoreras.
-15	Avsluta (SIGTERM). Standardsignal från kill. Denna signal kan fångas (trap) av processen, som kan avsluta sig själv på ett kontrollerat sätt.

NAMN

l - listar information om filerna och innehållet i biblioteken.

SYNTAX

l [-VRCxmlnogtasderucpFbqifP[E]1A] [namn ....]

FUNKTION

Kommandot *l* är exakt samma som kommandot *ls* givet med tillvalet **-l**, *ls* **-l**.

Tillvalet **-1** bevaras även om andra tillval lägges till.

Se beskrivningen av kommandot *ls*.

NAMN

labelit - skapa/visa etikett för filsystemet

SYNTAX

/etc/labelit [-V] filsystem [fsysname [volname]]

FUNKTION

Kommandot *labelit* används för att skriva en etikett på ett filsystem eller visa etiketten på ett befintligt filsystem. Filsystem är namnet på den yttre enhet eller fil där det filsystem finns på vilket en etikett skall skrivas, alternativt läsas.

Etiketten är uppbyggd av två stycken sex tecken långa fält: fsysname som innehåller namnet på filsystemet och volname som innehåller volymnamnet.

Om ingen etikett specificeras kommer *labelit* att visa informationen om filsystemet, inklusive eventuell etikett. Om en etikett specificeras kommer den att skrivas till filsystemet.

TILLVAL

-V Skriver ut versionsnumret för kommandot *labelit*

FILER

/..systemfiles/volumedescr

HÄNVISNING

Ingen

FELMEDDELANDEN

Inga

ANMÄRKNING

För att kunna skriva en etikett med *labelit* till ett filsystem som är mountat måste användaren ha skrivrättighet till filen **/..systemfiles/volumedescr** på just det filsystemet. För att skriva en etikett till ett filsystem som inte är mountat, krävs skrivrättighet till den yttre enheten eller filen där filsystemet finns. För att läsa de etiketter som finns på ett filsystem som inte är mountade, krävs läsrättigheter till enheten eller filen. För att läsa etiketter på ett filsystem som är mountat krävs dock inga läsrättigheter.

labelit kan inte användas på bandstationer.

LABELIT(1M)

NAMN

lc - listar innehållet i bibliotek i flera kolumner.

SYNTAX

lc [-VRCxmlnogtasderucpFbqifP[E]1A] [namn ....]

FUNKTION

Kommandot *lc* är samma som kommandot *ls*, givet med tillvalen **-Cc**, dvs. *ls* **-Cc**. Tillvalet **-Cc** bevaras även om andra tillval lägges till.

Se beskrivningen av kommandot *ls*.

LC(1)

NAMN

line - läs och skriv en rad.

SYNTAX

line [-V]

FUNKTION

line läser en rad (till ett new-line-tecken) från standard input och skriver raden till standard output. Det returnerar utgångsstatus 1 om filslut påträffas och skriver alltid ut minst new-line tecknet. Det används ofta i shellprocedurer för att läsa från användarens terminal.

TILLVAL

-V Skriver ut versionsnumret för kommandot *line*

FILER

Inga

HÄNVISNING

sh(1)

LINE(1)

NAMN

ln - skapar en länk mellan filer.

SYNTAX

ln [-V] [-f] filnamn1 [filnamn2]

FUNKTION

Kommandot *ln* skapar en länk till den existerande filen filnamn1. Länken får namnet filnamn2. Om filnamn2 inte anges, får länken samma namn som sista delen av filnamn1 och länken placeras i det aktuella biblioteket. Om filnamn2 är ett biblioteksnamn, placeras länken i detta bibliotek, med namnet som den sista delen av filnamn1. Om filnamn2 är en existerande fil förstörs det gamla innehållet.

En länk (=ett filnamn) är en post i ett bibliotek som refererar till en fil. En fil kan ha flera länkar till sig. All information om filen kommer att behållas, såsom storleken på filen, åtkomsttillstånden, osv. Alla länkar till en fil är likvärdiga, även den som först bildades när filen skapades.

Alla förändringar av filens innehåll är helt oberoende av antalet länkar till filen. Däremot tas inte filen bort ur filsystemet förrän samtliga länkar tagits bort.

Information om hur många länkar som finns och vilka filer som är länkade med varandra kan fås med kommandot *ls*.

Om åtkomsttillstånden för filen filnamn2 inte tillåter skrivning, stoppas kommandot och filnamn2 och dess åtkomsttillstånd skrivs ut på standard output. Systemet väntår därefter på ett svar från standard input, innan det kan fortsätta. Exekveringen fortsätter om svaret börjar med y och användaren har rättighet att ta bort filnamn2. Om inte avbryts kommandot. Om tillvalet **-f** ges eller om standard input inte är en terminal, ställs ingen fråga och länken skapas, om möjligt.

Anmärkning: Det är förbjudet att göra en länk till ett bibliotek eller att skapa länkar mellan olika filsystem.

TILLVAL

-V	Skriver versionsnummer till kommandot *ln*
-f	Skapar länken utan frågor om det är tillåtet, även om inget skrivtillstånd finns till filen filnamn2.

FILER

Inga

EXEMPEL

Exempel 1:

```
ln /pal/hty/chap1
```

Skapar en länk i aktuellt bibliotek till filen chap1 i biblioteket **/pal/hty**. Den nya länken får filnamnet chap1.

Exempel 2:

```
ln /pal/hty/chap5 kap5
```

Skapar en länk i aktuellt bibliotek till filen chap1 i biblioteket **/pal/hty**. Det nya filnamnet är kap5. Båda namnen har samma statusnivå, och båda namnen kan användas.

HÄNVISNING

cp(1), ls(1), mv(1)

NAMN

load - visar CPU:ns genomsnittsbelastning

SYNTAX

load

FUNKTION

load-kommandot visar genomsnittsbelastningen under den senaste minuten, de senaste 5 och 15 minuterna. Om *load* startas av *init*(1) kommer den att fungera som en demon och beräkna medelvärdet av belastningen, annars visar *load* medelbelastningen.

FILER

/usr/adm/load Nuvarande belastningsvärden

HÄNVISNING

init(1)

NAMN

login - inloggning till systemet

SYNTAX

exec login [-V] [-L *loginstr*] [-P *passtr*] [-D *dialstr*] [användarnamn]

FUNKTION

Kommandot *login* används för att logga in i systemet och låter användaren identifiera sig för systemet. *login* kan exekveras som ett kommando av en användare eller av systemet så snart kontakt etablerats. *login* exekveras också av systemet när en tidigare användare terminerat sin ursprungliga login-shell genom att skriva CTRL-D, vilket normalt är tecknet för 'end-of-file'.

Om *login* används som kommando måste det ersätta den ursprungliga kommandotolken, som startades då användaren loggade in till systemet. Detta görs från ursprungliga shell med kommandot *exec*, t.ex:

```
exec login
```

login frågar efter användarnamn (om detta inte angivits som argument), och, om detta är krävs, också lösenord. Ekot slås av, när så är möjligt, medan lösenordet skrivs för att det inte skall visas på bildskärmen.

Tillval **-L, -P** och/eller **-D** kan användas för att definiera egna textsträngar som frågor för användarnamn, lösenord och sekundärt lösenord (dial-up password). Som standard blir frågorna: login:, Password: och Dialup password: Ingen loginförfrågan ges om användarnamnet angivits som argument till kommandot *login*. Vid systeminitierad *login* läser kommandot *getty* användarnamnet innan det startar kommandot *login* och eventuella textsträngar för tillvalen **-L, -P** och **-D** kan läggas in i filen **/etc/gettydefs.**

I vissa installationer, kan ett tillval vara aktiverat som kräver att man anger ytterligare ett lösenord (dial-up). Detta händer bara vid uppringda anslutningar, och då efterfrågas Dialup password:. Då måste man ange både vanligt lösenord och ett dialup lösenord för att kunna logga in.

Detta andra lösenord krävs enbart om hela pathname för den använda terminalporten (t ex **/dev/tty05** eller **/dev/pk00** eller liknande) specificeras på en rad i filen **/etc/dialups** och användarens lösenord för uppringning måste specificeras i filen **/etc/d_passwd**. Fälten i **/etc/d_passwd** har samma form som i den vanliga filen **/etc/passwd**.

Fälten är användarnamn, valfritt dialup lösenord och korrekta siffror för användar-ID och grupp-ID måste anges korrekt fram till och med kolon efter grupp-ID. Nedan är ett exempel på en rad i **/etc/d_password** där inget lösenord är definierat:

```
kalle::114:1: Återstod av raden som ignoreras
```

Lösenordet för uppringning (dialup) kan definieras och ändras på samma sätt som ett vanligt lösenord, dvs med kommandot *passwd*, men nu med tillvalet **-f**. Exempel:

```
passwd -f/etc/d_passwd kalle.
```

Av säkerhetsskäl är det mycket viktigt att olika lösenord används i **/etc/passwd** och **/etc/d_passwd**. Normalt definierar systemadministratören lösenord för uppringning.

Observera: Fältet för lösenord i **/etc/d_passwd** får inte innehålla åldringskoder. Jämför beskrivningen för kommandot *passwd*.

Om man inte lyckas med inloggningen inom viss tid (ca: en minut) avslutas inloggningen och eventuell uppringd förbindelse kopplas ner.

Efter genomförd login, uppdateras vanligen loggfilerna, proceduren **/etc/profile** genomförs, dagens meddelande **(/etc/motd)**, om något sådant finns, skrivs ut; användar-ID, grupp-ID, inloggningsbiblioteket, och kommandotolken (vanligen *sh*) initieras. Filen **.profile**, om sådan finns i inloggningsbiblioteket, bearbetas. Dessa specifikationer för användaren finns på motsvarande rad i filen **/etc/passwd**.

Ett fullständigt pathname för kommandotolken anges i sista fältet i **/etc/passwd**. Om fältet är tomt används som standard kommandotolken */bin/sh*. Processnamnet på kommandotolken blir den sista delen av filnamnet, och föregås av ett bindestreck (-) för att ange att det startas med login som ursprunglig kommandotolk (t ex **-sh**).

Om sista fältet i **/etc/passwd** är '*', ändras root-biblioteket till det bibliotek som angivits som inloggningsbibliotek. Därvid exekveras *login* på nytt på den nya nivån som måste ha en egen root-filstruktur med */bin/login* och **/etc/passwd** och länkar eller kopior på alla kommandon och filer som ska användas, liksom länkar till enheterna i biblioteket **/dev**.

Omgivningen initieras bl a till:

```
LOGNAME=ditt-användar-namn
HOME=inloggningsbiblioteket          (från /etc/passwd)
PATH=:/bin:/usr/bin
SHELL=sista-fältet-på-passwd-raden   (från /etc/passwd)
MAIL=/usr/mail/användarnamn
TZ=tidszon för lokal tid        (från /etc/timezone)
```

För super-user (root) ingår även **/etc** i variabeln PATH.

Variabeln SHELL definieras inte om motsvarande fält i **/etc/passwd** är tomt. Det är proceduren **/etc/profile** som visar dagens meddelande, om sådant finns. Omgivningen kan ändras med användarkommandon i filen **.profile**

Omgivningen kan också utvidgas eller ändras genom att ange fler argument till *login*, antingen vid start med *exec login* ..... eller vid login-frågan efter användarnamnet. Login-argumenten kan antingen vara av formen xxx eller YYY=xxx. Argument utan likhetstecken placeras i

omgivningen som

```
Ln=xxx
```

där n är ett nummer som startar med 0 och ökas med 1 varje gång ett nytt variabelnamn behövs. Variabler som innehåller tecknet = placeras i omgivningen utan ändring. Om de redan finns i omgivningen ersätts de gamla värden. Det finns två undantag. Variablerna PATH och SHELL kan inte än-

dras på detta sätt. Detta är till för att hindra någon, som loggar in med en restriktiv shell att skapa sekundära shell som inte är begränsade.

Både *login* och *getty* förstår vissa enkla citeringsregler. Specialtecken kan skrivas in genom att ange tecknet \ före specialtecknet. På detta sätt kan till exempel ett MELLANSLAG eller ett TAB-tecken skrivas in.

Vissa extra-parametrar för kommandot *login* kan anges i kommentarfältet på användarens rad i filen **/etc/passwd**. Dessa är 'unique', 'pri-nn' eller 'console'. Om fler än en anges måste de separeras med ett kommatecken (,).

unique	Ingen annan användare kan logga in med samma namn.
pri=nn	Användarens prioritet minskas med det decimala värdetnn (0 - 20), såsom beskrivits för kommandot nice. För systemkonsolen enbart minskas prioriteten automatiskt med 16 för vanliga användare, men inte för super-user (root).
console	Denna användare kan endast logga in på systemkonsolen. Används ofta för super-user (root).

TILLVAL

-V	Skriver ut versionsnumret för kommandot *login*.
-L *str*	Definierar en textsträng som används för att fråga efter användarnamnet.
-P *str*	Definierar en textsträng som används för att fråga efter lösenordet.
-D *str*	Definierar en textsträng som används för att fråga efter lösenordet för uppringning (dialup).

FILER

/etc/utmp	loggfil
/etc/wtmp	loggfil
/usr/mail/ditt-namn	mailbox för användarnamn
/etc/passwd	lösenordsfil
/etc/profile	systemets profil
.profile	användarens loginprofil
/etc/dialups	dialup terminallinjer
/etc/d_passwd	dialup lösenordsfil
/etc/motd	dagens meddelande
/etc/timezone	lokal tidszon
/etc/gettydefs	valfria frågesträngar

HÄNVISNING

getty(1M), mail(1), newgrp(1M), passwd(1), sh(1), su(1M)

FELMEDDELANDEN

Vid fel, visas ett felmeddelande som förklarar felet. Några felsituationer visas nedan:

- Om användarnamn och lösenord inte matchar varandra.

- Om login inte kan öppna filen **/etc/passwd** eller när inget användarbibliotek har specificerats.
- Om man försöker att exekvera *login* som ett kommando utan att använda shell's intern-kommando *exec* eller från något annat shell.

ANMÄRKNING

Tecknen för ERASE och KILL är # och @ när man loggar in med användarnamnet och lösenordet. De ändras ofta efter login till de normala värdena i D-NIX: CTRL-H för ERASE och CTRL-X för KILL. ERASE tar bort ett tecken vid inmatning och KILL tar bort en hel rad vid inmatning från terminalen.

Observera: Om en vanlig användare loggar in på systemkonsolen minskas automatiskt prioriteten med värdet 16.

NAMN

logname - ger användarens login-namn

SYNTAX

logname [-V]

FUNKTION

logname skriver ut användarens login-namn på standard output. Namnet tas från shell-variabeln $LOGNAME.

TILLVAL

-V Skriver ut versionen för kommandot *logname*

FILER

Inga

HÄNVISNING

login(1), sh(1), tty(1)

FELMEDDELANDEN

Inga

LOGNAME(1)

NAMN

lp, cancel - skickar/avbeställer utskriftsbegäran till LP.

SYNTAX

lp [-V] [-c] [-d*dest*] [-m] [-n*antal*] [-o*tillval*][-s] [-t*titel*] [-w] filer

cancel [-V] [id ...] [skrivare ....]

FUNKTION

Angivna filer med tillhörande information benämns en utskriftsbegäran (request). Kommandot *lp* styr och effektuerar denna utskriftsbegäran till en radskrivare. Om inga filer anges sker utskrift från standard input. Filnamnet '-' står för standard input och kan ges på kommandoraden tillsammans med filer i övrigt. Filerna skrivs ut i den ordning de anges i kommandot.

lp skapar ett unikt id-nummer för varje utskriftsbegäran och skriver ut detta på standard output. Id-numret kan användas för att avbeställa (se *cancel*) eller för information om status på en utskriftsbegäran. (se *lpstat*).

Observera: Det finns två olika spooler-system för skrivare i D-NIX, *lp*-systemet och *lpr*-systemet. Endast ett av dem får vara aktivt.

Följande tillval för *lp* kan ges i godtycklig ordning och blandas med filnamn.

-V Skriver ut versionsnumret för kommandot *lp* eller *cancel*

-c Gör omedelbart kopior av de filer som skall skrivas ut. Normalt kopieras annars inte filer före utskrift utan länkas när så är möjligt. Om tillvalet **-c** inte anges måste användaren se till att filerna inte tas bort innan de har skrivits ut i sin helhet. Det bör också observeras att, i frånvaro av tillvalet **-c**, alla ändringar gjorda i en fil sedan utskriftsbegäran gjordes, men före utskrift, kommer att påverka utskriften.

-d*dest* Välj med *dest* den logiska skrivare eller den klass av skrivare på vilka utskrift skall ske. Om *dest* är en skrivare kommer utskriften att ske endast till denna. Om *dest* är en klass skrivare kommer utskriften att ske på första tillgängliga skrivaren inom klassen. Under vissa förhållanden (skrivaren inte tillgänglig, begränsat filutyrymme, etc) kan en utskriftsbegäran bli avvisad. Se *accept* och *lpstat*. Normalt tas *dest* från shellvariabeln LPDEST om denna är satt. Annars tas standarddestinationen från systemet (om det finns någon). Destinationsnamnen varierar i olika system och även skrivare på andra datorer, anslutna via nätverk, kan anges. Se *lpstat* .

-m Skicka meddelande med *mail* sedan filerna har skrivits ut. Som standard skickas inget meddelande vid normal

avslutning av en utskriftsbegäran. Tillvalet kan endast användas om kommandot *mail* är tillgängligt i systemet.

-n*antal* Skriv *antal* kopior av utskriften. Standardantal är 1.

-o*tillval* Specificerar skrivarberoende eller klassberoende tillval. Flera sådana tillval kan ges genom att specificera nyckeln **-o** (bokstaven o) mer än en gång. För information om giltiga tillval, se Modell-program i *lpadmin*.

-s Undertryck meddelanden från *lp* , såsom "request id is....." ("ID på utskriftsbegäran .....").

-t*titel* Skriver ut en *titel* med stora bokstäver (banner) på första sidan.

-w Skriver ett meddelande på användarens terminal sedan filerna skrivits ut. Om användaren inte är inloggad sänds ett meddelande istället med *mail*.

Kommandot *cancel* avbeställer en utskriftsbegäran som givits med kommandot *lp*. Argumenten i kommandot kan vara antingen ett id-nummer, enligt det meddelande som givits av *lp* - eller ett skrivarnamn. För att få en lista på tillgängliga id-nummer och namn kan kommandot *lpstat* användas. När ett id-nummer anges avbeställs tillhörande utskriftsbegäran även om utskrift pågår. Om ett skrivarnamn anges avbeställs den utskriftsbegäran som är under utskrift på den skrivaren. I båda fallen innebär en avbeställning av en pågående utskrift, att nästa eventuellt köande utskriftsbegäran kommer att börja skrivas ut.

Om en fjärransluten skrivare (via ett nätverk) används kommer *cancel* att utföras i den fjärranslutna datorn.

FILER

/usr/spool/lp/*

Se *lpadmin* för en fullständig fil-lista.

HÄNVISNING

cancel(1), enable(1), disable(1), lpstat(1), accept(1M), lpadmin(1M), lpsched(1M), mail(1)

NAMN

lpadmin - konfigurerar LP kösystem

SYNTAX

/usr/lib/lpadmin [-V] -p*skrivare* [tillval]

/usr/lib/lpadmin [-V] -x*dest*

/usr/lib/lpadmin [-V] -d[*dest*]

FUNKTION

lpadmin konfigurerar skrivarsystemet LP genom att definiera logiska skrivare, klasser och fysiska enheter. Det används för att lägga till och ta bort destinationer, ändra klasstillhörighet, ändra skrivarenheter, ändra interfaceprogram (filter) och för att ändra systemets standard-destination. *lpadmin* får inte användas när processen *lpsched*, köhanteringsprogrammet för LP, arbetar utom i de fall där detta anges nedan.

Genom att logga in som användaren *lp,* kan kommandona för lp-systemet ges utan full sökväg (pathname) eftersom PATH då definieras i filen **/usr/spool/lp/.profile**. Hem-biblioteket blir också **/usr/spool/lp.**

TILLVAL

Ett, och endast ett, av tillvalen **-p, -d** eller **-x** måste anges för att kommandot *lpadmin* skall kunna utföras på ett korrekt sätt.

-V Skriver ut versionsnumret av kommandot *lpadmin*.

-d[*dest*] Anger *dest* som systemets standard-destination. Den måste existera. Om *dest* inte anges, får systemet ingen standarddestination. Detta tillval kan användas även om *lpsched* håller på att arbeta. Inga andra tillval är tillåtna tillsammans med **-d**.

-x*dest* Tar bort destinationen *dest* från LP-systemet. Om *dest* är en skrivare som är enda medlemmen av en klass kommer klassen också att tas bort. Inga andra tillval är tillåtna med **-x.**

-p*skrivare* Anger namnet på en logisk skrivare till vilken alla tillval nedan hänför sig. Om *skrivare* inte finns som namn kommer det att skapas. Följande tillval är användbara endast med **-p** och kan förekomma i godtycklig ordning.

-c*klass* I Infogar skrivaren *skrivare* i den angivna klassen *klass*. Om klassen inte finns kommer den att skapas.

-e*printer* Kopierar en befintlig skrivares interface-program till att gälla för *skrivare*.

-h Anger att den enhet *skrivare* står för är direktansluten. Detta tillval förutsättes när en nylogisk skrivare skapas såvida inte tillvalet **-l** anges.

-i*interface*	Definierar ett nytt interface-program för *skrivare*. *interface* är filnamnet med det nya programmet.
-l	Anger att enheten, dit *skrivare* skriver, är en login-terminal. Köhanteringskommandot *lpshed* deaktiverar (*disable*) alla login-terminaler varje gång det startas. Innan *skrivare* aktiveras (*enable*) måste tillhörande enhet definieras med *lpadmin*.
-m*model*	Väljer ett standardprogram *model* som interface-program för *skrivare*. Model är ett av de programnamn som levereras med LP. (se Modell-program nedan).
-r*klass*	Tar bort skrivaren *skrivare* från angiven *klass*. Om *skrivare* är den enda skrivaren i klassen tas även klassen bort.
-v*enhet*	Definierar en ny enhet som skrivare *skrivare* ska skriva till. *enhet* är pathname till en fil eller fysisk enhet, till vilken LP har skrivtillstånd. Observera att inget hindrar att samma enhet används av flera logiska skrivare. Om endast tillvalen **-p** och **-v** anges, kan *lpadmin* användas även medan *lpshed* är aktivt. Standard output från interface-programmen blir automatiskt riktade till denna enhet. För fjärr-anslutna skrivare används normalt inte enhetsparametern, vilken då normalt bör sättas till **/dev/null**.

BEGRÄNSNINGAR

- När man skapar en ny skrivare måste tillvalet **-v** samt ett, och endast ett, av tillvalen **-e**, **-i** eller **-m** anges.
- Av tillvalen **-h** och **-l** kan endast ett anges.
- Namn på logiska skrivare och klasser får inte vara längre än 14 tecken och bestå endast av tecknen A-Z, a-z, 0-9 och understrykningstecknet (_).

MODELL-PROGRAM

Standard interface-program (model) för skrivare levereras med LP-systemet. Dessa är shell-procedurer vilka fungerar som filter mellan *lpshed* och de fysiska enheterna. Alla modell-program finns i biblioteket **/usr/spool/lp/model** och kan användas som de är i kommando-sekvensen *lpadmin* **-m**. Modell-program skall ha åtkomsttillstånden 644 om de ägs av *lp* och bin och 664 om de ägs av bin och bin.

LP-administratörer kan modifiera en kopia av ett modell-program och använda denna med *lpadmin* **-i** för användning med en viss skrivare.

Nedan följer några exempel på modellprogram och de tillval som kan ges på kommandot för *lp*-tillvalsbokstaven **-o**:

dumb	Interface för en radskrivare utan speciella funktioner och protokoll. Form feed förutsättes. Detta är en bra

modell att kopiera och ändra för skrivare utan modell-program.

1640 DIABLO 1640 terminal-skrivare, 1200 baud och XON/XOFF-protokoll.

Tillval:

-12 12 tkn/tum (10 tkn/tum är standard)

-f Använd inte filter 450(1). Utdata har redan behandlats av antingen 450(1) eller av nroff(1) med 450-tabellen.

hp Hewlett-Packard 2631A radskrivare, 2400 baud.

Tillval:

-c Komprimerad skrift.

-e Expanderad skrift.

prx Printronix P300 eller P600 skrivare som använder XON/XOFF protokoll, 1200 baud.

rlp Modellprogram för fjärransluten skrivare, via D-NET eller Ethernet LAN. **Observera:** Parametern SERVER skall anpassas till användarens system.

FILER

/usr/spool/lp/*	Bibliotek för LP-filer.
/usr/spool/lp/.profile	Filen **.profile** för användare *lp*.
/usr/spool/lp/class/*	Filer för varje klass med en lista av skrivare som hör till varje klass.
/usr/spool/lp/default	Innehåller namnet på standard-skrivaren.
/usr/spool/lp/interface/*	Shell-procedurer för varje definerad skrivare i systemet.
/usr/spool/lp/log	*lpsched* loggfile.
/usr/spool/lp/member/*	En fil för varje skrivare, som anger den fysiska enheten. Används ej för fjärranslutna skrivare.
/usr/spool/lp/model/*	En samling standard inter-face-procedurer för skrivare. Se separat beskrivning.
/usr/spool/lp/remote/*	Speciella program för hantering av fjärranslutnaskrivare.
/usr/spool/lp/SCHEDLOCK	Temporär fil, används av *lpsched*.

EXEMPEL

Exempel 1:

Antag en existerande Hewlett-Packard 2631A radskrivare benämnd hp2. Ett hp modell interface kommer att användas efter kommandot:

```
/usr/lib/lpadmin -php2 -mhp
```

Exempel 2:

För att erhålla komprinerad skrift på hp2, använd kommandot:

```
lp -dhp2 -o-c filer
```

Exempel 3:

En skrivare DIABLO 1640 kallad st1 kan läggas till LP-systemet med kommandot:

```
/usr/lib/lpadmin -pst1 -v/dev/tty20 -m1640
```

Exempel 4:

Följande kommando skriver ut filen **passwd** på st1 med 12 tkn/tum.

```
lp -dst1 -o12 /etc/passwd
```

Observera: Tillvalet **-12** för modellen 1640 skall aldrig användas tillsammans med *nroff*(1).

HÄNVISNING

accept(1M), cancel(1), disable(1), enable(1), lp(1), lpsched(1M), lpstat(1), reject(1M)

ANMÄRKNING

Genom att logga in som användare *lp* kan kommandona i *lp*-systemet ges utan full sökväg. Då blir hembiblioteket **/usr/spool/lp**, och PATH definieras i filen **/usr/spool/lp/.profile**.

NAME

lpd - hanterar utmatning till skrivare från skrivarspoolern lpr.

SYNTAX

/usr/lib/lpd [-V]

FUNKTION

lpd skriver ut utskriftsbegäran från kommandot *lpr* på radskrivare.

Kommandot används aldrig av normala användare eftersom kommandot *lpr* automatiskt startar upp processen *lpd*. Vid systemstart startas *lpd* automatiskt för att skriva ut eventuella filer lämnade i kön när systemet sist stängdes av.

Se kommandot *lpr* för detaljinformation.

OPTIONER

-V Skriver ut versionsnumret för kommandot *lpd*.

FILER

/usr/spool/lpd/* /etc/rc

HÄNVISNING

lpr, queue

LPD(1)

NAMN

lpr (line printer spooler) - spooler för skrivare

SYNTAX

lpr [-V -r -c -s -f -m*user* -n*k* -p*n* -h*n* -t*type*] [filnamn]

FUNKTION

Kommandot *lpr* skickar filer till en utskriftskö. Om ingen fil anges läser kommandot från standard input.

Observera: I grundsystemet finns två olika spoolersystem, *lp*-systemet och *lpr*-systemet. Endast en kan användas åt gången.

Om flera skrivare används väljer systemet lämplig fysisk skrivare beroende på villkoren i **-t** tillvalet som jämförs med strängvillkoren i filen **/usr/lib/lpdpar/lpdtable.** Inledande och avslutande filer kan automatiskt sändas till skrivarna. Filterprogram kan också användas automatiskt för att styra speciella kontrollsekvenser eller grafikprogram, särskilt konfigurerade för varje skrivare. Speciella initieringskommandon kan väljas automatiskt.

För att återstarta utskrift av eventuella filer i utskriftskön efter system start, 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 &
```

TILLVAL

-V	Skriver ut versionsnumret för kommandot *lpr*.
-r	Tar bort filen efter det att den ställts i kö för utskrifter.
-c	Kopierar filen för att förhindra att några ändringar sker innan utskriften.
-s	Skriver ut filerna i den ordning de är sända till skrivar spoolern. Som standard kan systemet välja att skriva ut de mindre filerna före de större.
-f	Anger formaterad utskrift, dvs. gör TAB-tecken till mellanrumssträngar. TAB har det oktala ASCII-värdet 011. TAB positioner förutsätts på raden i var 8:de kolumn. **-f** tillvalet möjliggör också användandet av filterprogram, valda ur biblioteket **/usr/lib/lpdpar/lpdhandler.**
-m*user*	Rapporterar via *mail* när utskiften är klar. Strängen *user* är användarnamnet till vilket meddelandet skall sändas. Detta tillval fungerar endast om kommandot *mail* är tillgängligt i systemet.

-n*k*	Skriver ut *k* antal kopior av filen.
-p*n*	Sänker den dynamiska utskriftsprioriteten med *n*. Endast för super-user kan *n* vara negativt. Prioriteten styr utskriftsordningen av filerna i utskriftskön. Standardprioriteten är 40. Prioriteten kan skrivas ut med kommandot *queue* . Processprioriteten hos skrivarspoolern i systemet påverkas inte.
-h*n*	Sätter antalet titelsidor (banners) till *n*. En titelsida är en sida som har användarnamnet, datum och tid skrivet med mycket stora bokstäver. Standard är att en titelsida skrivs ut före datainnehållet.
-t*typ*	Specificerar vilken *typ* av skrivare som skall användas. Detta tillval gäller före shell-variabeln PRTYPE som kan ha en standard-typ sträng. Den fysiska skrivaren väljs genom att jämföra strängen/arna i *typ* med strängarna i filen **/usr/lib/lpdpar/lpdtable**. *typ* kan bestå av flera strängar, separerade med kommatecken (,). Valsträngen default i tabellfilen är reserverad och anger standardskrivare som ska användas om varken tillvalet **-t** eller variabeln PRTYPE är definierade.

FILER

/usr/spool/lpd/*	Temporära köfiler.
/usr/lib/lpd	Spooler program.
/usr/lib/lpdpar/lpdtable	Skrivarvals tabell.
/usr/lib/lpdpar/devinit/*	Initieringsprogram.
/usr/lib/lpdpar/lpdheader/*	Titelfiler.
/usr/lib/lpdpar/lpdhandler/*	Filtreringsprogram.
/usr/lib/lpdpar/lpdtailer/*	Avslutningsfiler.

EXEMPEL

Exempel 1:

```
lpr -r myfile
```

Skickar filen **myfile** till utskriftskön, och tar därefter bort den ur aktuellt bibliotek. Standardskrivaren väljs enligt typsträngen i PRTYPE. Om ingen PRTYPE är definerad, väljs den första lediga skrivaren som har valsträngen 'default' i tabellfilen.

Exempel 2:

```
lpr -c myfile
```

Skapar en kopia av **myfile** till spoolkön, detta sker för att inga ändringar skall ske i dokumentet före utskrift.

Exempel 3:

```
ls -l | lpr -h0
```

Skickar resultatet av *ls* via en pipe till *lpr* som i sin tur lägger det i utskriftskön. Ingen titelsida skrivs ut.

Exempel 4:

```
lpr -tpretty mytext
```

Sänder filen *mytext* till spoolkön för utskrift från en skrivare med valsträngen pretty. Enligt exemplet nedan av filen **/usr/lib/lpdpar/lpdtable,** skrivs den ut på den fysiska skrivaren **/dev/lp2**.

Exempel 5:

```
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. Med tabellfilen nedan, används den skrivare som är ansluten till **/dev/lp3**.

Exempel 6:

```
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/lpdpar/lpdhandler** för den aktuella skrivaren, kommer det att användas. I varje fall ersätts TAB:ar i textsträngarna till mellanrumssträngar före utskrift. Med tabellfilen nedan används den skrivare som är ansluten antingen till **/dev/lp1** eller **/dev/lp3**.

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** tillvalet 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 behöver matchas vilket används för att ange en grupp av skrivare. Om flera skrivare uppfyller kraven vid valet, används den skrivare som är ledig för tillfället. En standard typsträng kan definieras i shellvariabeln PRTYPE. Om så inte är fallet, väljs den skrivare som är markerad med valsträngen 'default' i tabellfilen.

Om endast **/dev/lp** används i systemet och ingen initiering, inga titelfiler, avslutningsfiler eller filtreringsprogram används, är inte biblioteket **/usr/lib/lpdpar** nödvändigt.

Ett exempel på en tabellfil ges nedan.

Om en rad i tabellen börjar med en sträng, tolkas denna sträng som den fysiska utskriftsporten (enheten) och den andra strängen (tabellnamn) används för att välja initieringsprogram, titel- och avslutningsfilen och (**-f** tillval) filterprogram. Därpå följande strängar, såväl som tabellsträngen, används som val-strängar och jämförs med typsträngarna som givits i **-t** tillvalet.

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 i filen.

Om första raden i tabellfilen börjar med ett mellanrum eller en TAB, kommer alla strängar på denna rad att specificera valsträngar för vilka endast

en sträng behöver överensstämma. Se exempel 5 ovan och valexempel 3 nedan.

Exempel på en tabellfil med följande fält:

Device	Tabellnamn	Valsträngar
rum1	rum2	rum3
lp	facit4542	english medium default room1
lp1	facit4544	english graphic medium default room3
lp2	diablo	swedish pretty room1
lp3	graf2	english graphic room2

Denna tabell definerar **/dev/lp** som en Facit 4542 att vara standard skrivare (default) med medelhög hastighet och en engelsk teckenuppsättning. På samma sätt är **/dev/lp1** en Facit 4544 med en engelsk teckenuppsättning, grafisk presentation, medelhög hastighet och även den standard-skrivare. **/dev/lp2** är en diabloskrivare med en svensk teckenuppsättning. **/dev/lp3** är en andra skrivare för grafik. I detta tabellexempel, kan gruppvalsträngarna på första raden t ex beskriva den fysiska platsen för skrivarna.

Den andra strängen, kallad tabellnamn ovan, anger filnamn för titel- och avslutningsfiler, initieringsprogram och filtreringsprogram.

Om endast **/dev/lp** används i systemet och ingen initiering, inga titelfiler, avslutningsfiler eller filtreringsprogram används, behövs inte biblioteket **/usr/lib/lpdpar**.

Valexempel 1:

Användaren specificerar tillvalet -tenglish till kommandot *lpr* och utskriften dirigeras till första fria enhet som har 'english' som angiven teckenuppsättning. I detta fall gäller det både för **/dev/lp**, **/dev/lp1** och **/dev/lp3**.

Valexempel 2:

Skrivartypen kan också specifieras med en kombination av strängar, såsom -tenglish,graphic, vilket i detta fallet skulle betyda **/dev/lp1** eller **/dev/lp3**. Observera att kommatecken (,) krävs mellan varje sträng i tillvalet.

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 som enhetsnamnet, dvs. -tlp, -tlp1, -tlp2 eller -tlp3, i detta fall kan dock inga andra valsträngar 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 ett kommando med standard input och output dirigerad till skrivaren innan någon annan text skickas ut. Om denna fil inte finns, sker ingen initiering.

TITEL-/AVSLUTNINGSFILER TILL SKRIVARE (Bibliotek: lpdheader och lpdtailer)

Tabellnamn-strängen i tabellfilen är namnet på en fil i biblioteket **/usr/lib/lpdpar/lpdheader**. Data i denna fil skrivs alltid ut före annan text. Titelfilen kan innehålla strängar med skrivarkommandon för skrivaren. Dessa kan användas för att se till att skrivaren alltid skriver likadant. Om ingen fil finns med detta namn, skrivs initieringsdata ut.

Tabellnamn-strängen i tabellfilen är även namnet på en fil i biblioteket **/usr/lib/lpdpar/lpdtailer.** Data i denna fil skrivs alltid ut efter all annan text. Om ingen fil med detta namn finns, skrivs ingen särskild data ut i slutet av utskriften.

FILTERPROGRAM FÖR SKRIVARE (Bibliotek: lpdhandler)

Tabellnamn-strängen i tabellfilen är namnet på en fil i biblioteket **/usr/lib/lpdpar/lpdhandler.** Denna fil innehåller filterprogram, som används för avkodning av speciella sekvenser vid utmatning av data med grafisk presentation och andra speciella utskrifter. Detta är möjligt bara om **-f** tillvalet är givet och om filen med filterprogrammet existerar. Filterprogrammen läser från standard input (= data som skall skrivas ut) och skriver till standard output (blir automatiskt omdirigerad till skrivaren) Speciella sekvenser i datainnehållet startar och stoppar filterprogrammet.

0377	0377	typ	format	000	010

Den speciella sekvensen inleds av två bytes med de oktala ASCII-värdena 0377. Sekvensen ser ut som nedan och åtföljs av alla data som skall skrivas ut 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 fram till nästa speciella sekvens till filtret. För att avsluta filterprogrammet kan en sekvens sändas med typ som är 0, och format lika med strängen text.

Om typ är 1 kommer en eller flera binära block att följa den speciella sek-

id	storlek	binärdata

vensen. Filterprogrammet avslutas när ett binärt block som har sitt ID-fält lika med 1 är utskrivet.

Det binära blocket har följande format:

Fältet ID har normalt värdet 0 (ASCII 000) om flera block följer. Det sista binära blocket skall ha ID-värdet 1 (ASCII 001) för att filterprogrammet skall avslutas när alla binära data i blocket har skrivits ut. Fältet storlek är ett 16 bitar binärt värde som har mest signifikanta byte först. Fältet binärdata är en sträng med dimensionen storlek bytes och denna sträng sänds genom filterprogrammet till skrivaren.

HÄNVISNING

queue(1), mail(1)

FELMEDDELANDEN

lpr avbryts om en typ ges (**-t***typ*) som inte finns i tabellfilen **/usr/lib/lpdpar/lpdtable**.

NAMN

lpmove - flyttar utskriftsbegäran mellan olika destinationer i LP-systemet.

SYNTAX

/usr/lib/lpmove [-V] ids dest

/usr/lib/lpmove [-V] dest1 dest2

FUNKTION

lpmove flyttar en utskriftsbegäran (request) mellan olika destinationer. Kommandot får inte användas när *lpsched* är aktivt.

Den första formen av *lpmove* ovan flyttar en namngiven utskriftsbegäran till LP-destinationen dest. ids är ID-nummer för utskriftsbegäran givet av *lp*. Den andra formen flyttar alla utskriftsbegäran i kön från destinationen dest1 till destinationen dest2.

Kommandot beskrivs närmare tillsammans med kommandot *lpsched*.

HÄNVISNING

lpsched(1M), lpadmin(1M)

LPMOVE(1M)

NAMN

lppg - Delar upp en utskrift i sidor

SYNTAX

/usr/spool/lp/bin/lppg [-V] [-l *sidlängd*] id

FUNKTION

lppg är ett filterprogram som styrs av kommandot *lpsubmit* och används som extra filter i interfaceprogram till skrivare för att dela upp utskriften i sidor. *lppg* skapar två pipe-filer i **/tmp** ur filnamnet id, genom vilka det kommunicerar med *lpsubmit*. Användaren kan styra utskriften genom interaktiva kommandon. Detta beskrivs under kommandot *lpsubmit*.

Ett modell till ett interfaceprogram som använder *lppg* finns i filen **/usr/spool/lp/model/pager.**

Argumentet id är ett filnamn, normalt lika med ett könummer (request-id), och används av *lppg* för att skapa pipe-filerna.

TILLVAL

-V	Skriver ut versionsnumret för kommandot.
-l *sidlängd*	Antal rader på varje sida. Ingen sidindelning görs om tillvalet **-l** utelämnas. Då antas dokumentet redan vara sid-indelat med formfeed-tecken.

FILER

/usr/spool/lp/model/pager
/tmp/XXX-NNN.c
/tmp/XXX-NNN.a

HÄNVISNING

lp(1), lpsubmit(1)

LPPG(1)

NAMN

lpsched, lpshut, lpmove - startar/stoppar köhanteraren eller flyttar utskriftsbegäran i LP systemet.

SYNTAX

/usr/lib/lpsched [-V]

/usr/lib/lpshut [-V]

/usr/lib/lpmove [-V] ids dest

/usr/lib/lpmove [-V] dest1 dest2

FUNKTION

lpsched hanterar utskriftsbegäran (requests) från *lp* för utskrift på radskrivare. Kommandot används aldrig av normala användare. *lpshed* startas automatiskt vid systemstart. *lpsched* startas normalt med lägre prioritet som en bakgrundsprocess genom ett kommando i **/etc/rc**.

lpshut avbryter köhanteraren *lpsched*. Alla skrivare i verksamhet när lpshut ges kommer att sluta skriva ut. En utskriftsbegäran under utskrift när skrivaren stoppads med *lpshut* kommer att skrivas om från början när *lpsched* har återstartas. Alla LP-kommandon utför sina funktioner även när *lpsched* inte är aktiverat.

lpmove flyttar utskriftsbegäran (köade med *lp*) mellan olika destinationer. Kommandot får inte användas när *lpsched* är aktivt.

Den första formen av *lpmove* ovan flyttar en namngiven utskriftsbegäran till LP-destinationen dest. ids är det av *lp* givna id-numret för en utskriftsbegäran. Den andra formen flyttar alla utskriftsbegäran i kön för destinationen dest1 till destinationen dest2. Som en konsekvens av detta kommer också *lp* att avvisa alla nya utskriftsbegäran för dest1. (Se *reject*).

Observera att *lpmove* ignorerar eventuell reject-status för dest2 när flyttning sker till den nya destinationen. (Se *accept*).

TILLVAL

-V Skriver ut versionsnumret för kommandot.

FILER

/usr/spool/lp/*

Se *lpadmin* för en fullständig fil-lista.

HÄNVISNING

accept(1M), cancel(1), disable(1), enable(1), lp(1), lpadmin(1), lpstat(1), reject(1M)

LPSCHED(1M)

NAMN

lpshut - stoppar LP köhanterare *lpsched.*

SYNTAX

/usr/lib/lpshut [-V]

FUNKTION

lpshut stoppar köhanteraren (*lpsched*). Alla skrivare i verksamhet när *lpshut* ges kommer att sluta skriva ut.

Kommandot beskrivs närmare tillsammans med kommandot *lpsched.*

HÄNVISNING

lpsched(1M), lpadmin(1M)

LPSHUT(1M)

NAMN

lpstat - skriver ut status information för LP

SYNTAX

lpstat [-V -a[*lista*] -c[*lista*] -d -o[*lista*]-p[*lista*] -r -s -t -u[*lista*] -v[*lista*]] [ids]

FUNKTION

lpstat skriver ut information om skrivarspoolersystemet *lp*.

Om inga tillval ges skriver *lpstat* ut status för alla utskriftbegäran (requests) som gjorts med *lp* av användaren. Alla argument (ids) som inte är tillval förutsättes vara id-nummer (som ges av *lp*). *lpstat* skriver ut status för dessa. Om det angivna id-numret inte finns på den lokala datorn och motsvarande utskriftsbegäran sänts till en fjärransluten skrivare, startas automatiskt *lpstat* i den andra datorn och returnerar status information.

Tillval kan ges i godtycklig följd och kan blandas med olika argument. Några av tillvalen nedan kan följas av en lista av två slag: en lista på uppgifter utan mellanslag separerade med ett kommatecken, eller en lista på uppgifter inom citationstecken "..." separerade antingen med ett kommatecken eller med ett eller flera mellanslag. De följande två exemplen är likvärdiga:

```
-u"user1, user2, user3"
-uuser1,user2,user3
```

Då listan utelämnas efter tillvalsbokstaven ges all information som är relevant för tillvalet ifråga.

```
lpstat -o
```

skriver ut status för alla utskriftsbegäran.

TILLVAL

-V	Skriver versionsnumret för kommandot ***lpstat***.
-a[*lista*]	Skriver accept-status (relativt *lp*) för destinationer. *lista* är en lista på blandade skrivarnamn och klassnamn.
-c[*lista*]	Skriver klassnamn och medlemmar i klasserna. *lista* är en lista på klassnamn.
-d	Skriver systemets standard-destination för *lp*.
-o[*lista*]	Skriver status för utskriftsbegäran. *lista* är en lista på blandade skrivarnamn, klassnamn och ID-nummer på utskriftsbegäran.
-p[*lista*]	Skriver status för olika skrivare. *lista* är en lista på skrivarnamn.
-r	Skriver status på LP köhanterare *lpsched*.
-s	Skriver en statusöversikt, innefattande *lpsched* , systemets standard-destination, en lista på klassnamn och

deras medlemmar, samt en lista på skrivare med tillhörande fysiska enheter.

-t Skriver ut all statusinformation.

-u[*lista*] Skriver status för utskriftsbegäran från användare. *lista* är en lista på loginnamn.

-v[*lista*] Skriver namn på skrivare och de fysiska enheter som dessa riktas mot. *lista* är en lista med skrivarnamn.

FILER

/usr/spool/lp/*

Se *lpadmin* för en fullständig fil-lista.

HÄNVISNING

accept(1M), cancel(1), disable(1), enable(1), lp(1), lpadmin(1M), lpsched(1M), lpshut(1M), lpmove(1M), reject(1M)

ANMÄRKNING

Se kommandot *queue* om spoolsystemet *lpr* används i stället för LP-systemet.

NAMN

lpsubmit - Kontrollerar utskrift till skrivare

SYNTAX

lpsubmit [-V] id

FUNKTION

lpsubmit är ett kommando för att styra utskriften till en skrivare. Programmet är menystyrt, med följande kommandon i menyn:

- Starta utskrift
- Stoppa utskrift
- Skriv en sida
- Gå en sida framåt
- Gå en sida bakåt
- Gå till början av filen
- Avsluta men ta ej bort filen
- Avsluta och ta bort filen

Ett kommando väljs genom att man med hjälp av piltangenterna flyttar sig till det kommando man vill utföra och sedan trycker på **RETUR**-tangenten.

Programmet bygger på att man i interfaceprogrammet filtrerar utskriften genom kommandot */usr/spool/lp/bin/lppg*. I filen (**/usr/spool/lp/model/pager**) finns en modell för detta. När *lppg* startas i interfaceprogrammet kommer den att skapa en pipe-fil i **/tmp** från vilken den läser kommandon. När man ger ett kommando i *lpsubmit* så skrivs kommandot i denna pipe, vidare läser *lpsubmit* det aktuella sidnumret från en annan pipe i **/tmp**.

Argumentet id skall vara kö-numret (request-id) för den utskriftsbegäran som ska styras. Detta returneras av *lp* när utskriften startas.

TILLVAL

-V Skriver ut versionsnumret för kommandot *lpsubmit*

EXEMPEL

En utskrift kan gå till enligt följande:

```
$ lp -dpgs /etc/passwd
request id is pgs-1799 (1 file)
$ lpsubmit pgs-1799
```

(Nu skriver man ut listan genom att välja kommandon i menyn)

```
$
```

I detta fallet skriver man ut **/etc/passwd** på ett kontrollerat sätt. Den logiska skrivaren heter här pgs.

FILER

/usr/spool/lp/model/pager
/tmp/XXX-NNN.c
/tmp/XXX-NNN.a

HÄNVISNING

lp(1), lppg(1)

FELMEDDELANDEN

Ett felmeddelande returneras vid försök att styra utskrifter via logiska skrivare som ej använder *lppg*.

ANMÄRKNING

Endast aktuell utskrift kan skrivas ut, det går ej att få en annan utskrift att gå före i kön.

NAMN

ls - listar information om filer och innehållet i bibliotek

SYNTAX

ls [-VRCxmlnogtasderucpFbqifP[E]1A] [namn ...]

FUNKTION

Kommandot *ls* listar innehållet i de bibliotek som anges som argument och/eller listar den begärda informationen om de filer som anges som argument. Utskriften är i alfabetisk ordning om inget tillval anges, men innehållet i varje bibliotek blir utskrivet separat om inte tillvalet **-P** anges. Om kortformen *l* används kommer informationen alltid att listas som om *ls* **-l** var angivet (långt format).

Med **-C**, **-x** eller **-m** tillvalen förutsätts 80 teckens radlängd om inte radlängden finns definierad genom shellvariabeln COLUMNS eller som col-attribut i filerna **terminfo** eller **termcap**, adresserade genom shellvariabeln TERM.

Olika uppsättningar av standard-tillval kan defineras av användaren. Om shell variabeln LSW har satts till en *ls* tillvalssträng (t.ex. genom LSW=-lt), använder kommandot *ls* dessa som standardtillval. Ett annat sätt att definiera standard-tillval är att med andra namn skapa nya länkar till kommandot *ls*. Kommandot *ls* känner igen de länknamn som listas nedan. I detta fall kan andra tillval ges på kommandoraden som tillägg till standard-tillvalen.

Som exempel, skapar följande kommando en länk till *ls* med namnet *lf*. När kommandot *lf* används som en länk till *ls*, är standard-tillvalen **-CFc** för att lista filer i multipla kolumner med / eller * tillagt på slutet av namnen. Logga in som super-user innan länken skapas.

```
cd /bin
ln ls lf
```

De olika länknamnen och motsvarande standard-tillval är:

```
l       =>   ls -l
lc      =>   ls -Cc
lf      =>   ls -CFc
ll      =>   ls -lc
lm      =>   ls -mc
lr      =>   ls -Crc
lx      =>   ls -xc
ls      =>   ls med standard-tillval i $LSW
```

TILLVAL

-V Skriver ut versionsnumret av kommandot *ls*.

-R Listar alla underbibliotek rekursivt varefter de påträffas. För att kunna fungera med äldre system, accepteras tillvalet **-e** med samma funktion som **-R**.

-e Samma som **-R**.

-C Listning i flera kolumner med sortering nedåt i kolumnerna. Se kommentaren om radlängd ovan.

-x Listning i flera kolumner med sortering i sidled i raderna istället för nedåt. Se kommentaren om radlängd ovan.

-m Listar rakt över sidan och separerar filerna med komma-tecken. Se kommentaren om radlängd ovan.

-l Listar på långt format, dvs. följande information kommer att skrivas på skärmen, om den inte ändrats pga andra tillval. Se exempel 1 nedan för en fullständig förklaring till informationen.

* filtyp och åtkomsttillstånd, jämför kommandot *chmod*
* antal länkar
* ägare
* grupp
* storlek i bytes (eller major,minor nummer för specialfiler).
* tid för senaste uppdatering (=modification time). Jämför - **u** och **-c**.
* bibliotek/filnamn

Om filen är en specialfil, visar storleksfältet istället primära och sekundära enhetsnummer (major and minor device).

Tid för sista uppdatering har två olika former.

'Month Date Hour:Seconds' om nuvarande år eller

'Month Date Year' om äldre än 6 månader

-n Som -l tillvalet, men UID och GID skrivs på skärmen istället för de namnsträngar som hör till dem.

-o Som -l tillvalet, men gruppen visas inte.

-g Som -l tillvalet, men ägaren visas inte.

-t Sorterar listan enligt senaste uppdateringstid (= modification time), med den senast uppdaterade filen först. Detta tillval kan kombineras med **-u** för att sorteringen skall ske enligt sista åtkomsttiden eller med **-c** tillvalet för att sortera efter sista uppdateringstiden för i-noden.

-a Listar alla filnamn. De filer som har namn som börjar med punkt '.' kommer annars inte att listas.

-s Anger antalet 512-bytes block för varje fil/bibliotek, inklusive antalet indirekta block.

-d Namnet på biblioteket som angetts i kommandoargumentet skrivs ut på skärmen, normalt i förening

med **-l** tillvalet för att få status på filer/bibliotek, utan att lista innehållet i något underbibliotek.

-r Listan sorteras i omvänd ordning.

-u Använd senaste åtkomsttid istället för modifieringstiden, vid listning med **-l** tillvalet och/eller sortering med **-t** tillvalet.

-c Använd i-nodens senaste uppdateringstid istället för modifieringstiden vid listning med tillvalet **-l** och/eller sorterad med **-t** tillvalet. Detta kan exempelvis vara när åtkomsttillstånden ändrats.

-p Lägger till ett '/' efter varje filnamn som är listat om filen fungerar som bibliotek.

-F Lägger till ett '/' efter varje listat filnamn som är ett bibliotek, och lägger till en '*' efter varje filnamn om filen är en exekverbar fil.

-b Alla icke-grafiska tecken i filnamnen skrivs ut i oktal form som '\nnn', där nnn är det oktala ASCII värdet.

-q Alla icke-grafiska tecken i ett filnamn skrivs ut som fråge- tecken '?'.

-i Skriver filens i-nod nummer i första kolumnen i utskriften.

-f Varje argument som anges uppfattas som ett bibliotek, även om filtyperna inte indikerar detta. *ls* försöker tolka innehållet i biblioteken som filnamn samt listar filnamnen i den ordning de uppträder. Tillvalen **-R**, **-l**, **-t**, **-s** och **-r** tas bort och tillvalet **-a** används.

-P Skriv filnamnens sökväg i listningen, utgående från och inklusive bibliotek givna som argument.

-PE Samma som **-P** tillvalet, men inga felmeddelanden ges om användaren inte har läs- eller exekveringstillstånd i biblioteken som angivits på kommandoraden.

-1 Tvingar listning med en fil per rad genom att negligera tillvalen **C**, **m**, **l** och **x**.
Observera: Tillvalet är siffran ett (1), inte bokstaven l.

-A*n* Listar enbart filer med angivna åtkomsttillstånd. De tre databitarna i siffran *n* (en siffra 0..7) anger 'rwx' för läs/skriv/exekveringstillstånd. Om några bitar är satta i *n* listas enbart filer med tillstånd enligt dessa. För vanliga användare testas enbart användarens och gruppens tillstånd. För super-user anses alla filer ha läs/skrivtillstånd och enbart 'x'-tillståndet testas (bit 0 i *n*).

FILER

/etc/passwd	Användar-ID.
/etc/group	Grupp-ID.

EXEMPEL

Exempel 1:

```
ls -l
```

Listar innehållet av aktuellt bibliotek på långt format:

```
drwxrw-r--   3 karl     other      45  Nov 10 09:22 Manual
-rwxr-xr-x   1 oskar    other     189 Jan 18 11:09 kap1
-rw-r--r--   1 putte    other     258 Jan 22 14:32 kap2
```

Förklaringar, med biblioteket Manual som exempel:

d---------	Det första tecknet indikerar filtypen. p = Pipe (FIFO) specialfil, b = Block specialfil, d = Bibliotek, c = Teckenorienterad specialfil och - = Vanlig fil.
-rwxrwxrwx	Tre set med åtkomsttillstånd. För filägaren, gruppen och för alla övriga. r = Lästillstånd, w = skrivtillstånd och x = exekveringstillstånd. För bibliotek, x = åtkomsttillstånd till filer i biblioteket. Ett minustecken (-) ersätter bokstaven om motsvarande tillstånd inte gäller.
---S------	Set-user-ID. Ägarens ID sätts automatiskt vid exekvering.
---s------	Både set-user-ID och x tillstånd för filägaren.
------S---	Set-group-ID. Gruppen ID sätts automatiskt vid exekvering.
------s---	Både set-group-ID och x tillstånd för filgruppen.
---------T	Sticky bit. Används inte i D-NIX.
---------t	Både sticky bit och x tillstånd för övriga.
3	Antal länkar till filen Manual.
karl	Ägaren till filen Manual.
other	Gruppen till filen Manual.
64	Antalet tecken i filen. En biblioteksfil innehåller en tabell av filerna i biblioteket.
Nov 10	Sista modifieringstid (jämför **-u**, **-c**)
09:22	Sista modifieringstid, eller år om filen är äldre än 6 månader.
Manual	Filnamn. I detta fallet biblioteksnamnet.

Se kommandot *chmod* för detaljer om åtkomsttillstånd.

Exempel 2:

```
ls -lt
```

Listar innehållet sorterat enligt sista modifieringstid för filerna i det aktuella biblioteket.

```
-rwxr-xr-x    1 oskar    other      189 Jan 18 11:09 kap1
-rw-r--r--    1 putte    other      258 Jan 22 14:32 kap2
drwxrw-r--    3 karl     other      45  Nov 10 09:22 Manual
```

Exempel 3:

```
ls -a
```

Alla filer och bibliotek, även dolda, blir listade på följande sätt:

```
.  ( . = aktuellt bibliotek)
.. (.. = förälderbibliotek)
.profile
Manual
Kap1
Kap2
```

Exempel 4:

```
ls -ld Manual
```

Listar specificerat bibliotek utan att lista innehållet.

```
drwxr-xr-x 3 karl other 45 Nov 10 09:22 Manual
```

Exempel 5:

```
ls -i
1946 myfile
```

Ger bibliotekets/filens i-nodnummer. i-nodnumret är filens index-nummer vilket pekar ut var filen är lagrad på skivminnet.

Exempel 6:

```
ls -ut /Manual
```

Listar filer och bibliotek efter användningstid. Den fil eller bibliotek som senast användes skrivs ut först.

Exempel 7:

```
ls -f myfile
myfile
myfile:
total 0:
```

Försöker lista myfile som ett bibliotek. Om myfile skulle ha innehållt något, görs ett försök att lista innehållet som filer i ett bibliotek.

Exempel 8:

```
ls -g
-rw-r--r-- 1 övriga 58 Aug 7 09:58 myfile
```

Ger ett långt format av *ls*. Ägarens namn skrivs dock inte ut, endast gruppnamnet.

Exempel 9:

```
ls -lA1
-rwxr-xr-x 1 karl other 236 sep 15 14:24 mycmd
-r--r-xr-- 1 karl other 35  oct 12 10:10 myowncmd
```

Listar enbart filer för vilka användaren eller användarens grupp har lästillstånd. **Observera:** Första tecknet i -lA1 är lilla bokstaven l medan sista är siffran 1.

HÄNVISNING

chmod(1), find(1), l(1), lc(1)

FELMEDDELANDEN

Utgångsstatus från *ls* är normalt 0 om inget fel upptäcks. Vid fel sätts utgångsstatus enligt nedan, där flera fel ger en utgångsstatus som är summan av angivna värden.

Status	Beskrivning
1	Felaktiga tillval (usage)
2	Slut på minnesarean, ex för många parameterar
4	För långt filnamn
8	Angiven fil saknas
16	Fel i sökvägen (pathname)
32	Läsfel
64	Tillstånd saknas att gå in i valda bibliotek
128	Signal mottagen

ANMÄRKNING

ls kräver lästillstånd för ett bibliotek för att kunna lista innehållet i biblioteket. Dock inte om användaren har super-user privilegier.

Oskrivbara tecken i filnamnen kan förstöra utskriftsformatet eftersom kommandot *ls* inte tar bort dessa tecken om inte tillvalet **-b** eller **-q** anges.

NAMN

diab1130, diab 1320, diab 1420, diab2420,diab2430

ds90-10, ds90-11, ds90-20, ds90-21, ds90-30, ds90-30s, ds90-31,ds90-41

mc68k, mc68020, mc68030, mc68040

- returnerar sant (0) eller falskt (-1) och anger därmed nuvarande datortyp.

SYNTAX

ds90-10

-

-

-

mc68040

FUNKTION

Dessa kommandon returnerar en utgångsstatus som är 0 (=sann) enbart i det system som kommandonamnet anger eller i kompatibla system. Övriga kommandon returnerar en utgångsstatus skild från noll. Dessa kommandon används ofta i shellprocedurer och med utvecklingssystemets make-kommando för att skapa portabla program.

	Sann om datorn är en	
diab1130	- " -	diab1130
diab1320	- " -	diab1320
diab1420	- " -	diab1420
diab2420	- " -	diab2420
diab2430	- " -	diab2430
ds90-10	- " -	DS90-10
ds90-11	- " -	DS90-11
ds90-20	- " -	DS90-20
ds90-21	- " -	DS90-21
ds90-30	- " -	DS90-30
ds90-30s	- " -	DS90-30S
ds90-31	- " -	DS90-31
ds90-41	- " -	DS90-41
mc68k	Sann om processorn är en	Motorola 680x0
mc68020	- " -	Motorola 68020
mc68030	- " -	Motorola 68030
mc68040	- " -	Motorola 68040

TILLVAL

Inga.

EXEMPEL

```
if ds90-30 ; then echo 'I am a DS90-30';fi
if mc68030 ; then echo 'I use a M68030';fi
```

HÄNVISNING

sh(1), test(1), true(1), false(1).

MACHID(1)

NAMN

mail - sänd meddelande till användare eller läs meddelande.

SYNTAX

mail [-Vehpqr] [-f fil] [-F mottagare] **(läs meddelande)**

mail [-Vostw] mottagare **(sänd meddelande)**

FUNKTION

mail utan argument skriver ut meddelanden till användaren, ett i sänder i ordning sist inkommet, först utskrivet. För varje meddelande ställs en fråga med tecknet ? och en rad läses från standard input för ett kommando som anger vad som skall göras med meddelandet.

RETUR	Fortsätt med nästa meddelande
+ or n	Samma som RETUR
d	Ta bort meddelandet och fortsätt med nästa meddelande
d #	Ta bort meddelande nr #. Gå inte vidare till nästa meddelande
dq	Ta bort meddelande och avsluta *mail*
h	Visa ett fönster med brevhuvud runt nuvarande meddelande
h #	Visa brevhuvud för meddelande nr #
h a	Visa brevhuvuden för alla meddelanden i användarens *mailfil*
h d	Visa brevhuvuden för de meddelanden som valts att tas bort
p	Visa meddelandet igen.
-	Gå tillbaka till föregående meddelande.
a	Visa meddelande som ankom under det att du körde *mail*
#	Visa meddelande nr #
r [användare]	Svara avsändaren och annan/andra användare, ta sedan bort meddelandet
s [filer]	Spara meddelandet i angivna filer (mbox är standard).
y	Samma som tillvalet s
u [#]	Ta tillbaka borttaget meddelande nummer # (senast lästa är standard)
w [filer]	Spara meddelandet, utan brevhuvud, i angivna filer (mbox är standard).
m [mottagare]	Sänd meddelandet till angivna mottagare (du själv är standard).

q	Lägg tillbaka ej borttagna meddelanden i mail-filen och avsluta.
EOF (CTRL-D)	Samma som q.
x	Lägg tillbaka alla meddelanden i mail-filen oförändrade och avsluta.
!command	Gå in i en ny temporär shell för att utföra ett kommando.
*** eller ?**	Visa en kommandosammanfattning.

När mottagare namnges läser *mail* från standard input fram till ett end-of-file tecken(CTRL-D), eller till en rad som innehåller endast en punkt (.) och för in detta meddelande i varje mottagares *mail*-fil. Meddelandet föregås av ett brevhuvud med avsändarens namn och en adressmarkering. Rader som ser ut som adressmarkeringar i meddelandet, (exempelvis: "From xyz ...") föregås av tecknet >. Tillvalet **-t** gör att meddelandet föregås av namnen på alla mottagare som det skickas till.

En mottagare är vanligen ett användarnamn igenkännligt för login. Alternativt kan ett användarnamn i en annan dator anges, där denna är ansluten över något nätverk. Om en mottagare inte kan identifieras eller om *mail* avbryts under inmatning, sparas meddelandet i filen **dead.letter** för senare redigering och återsändning. Observera att filen **dead.letter** behandlas som en temporär fil eftersom den skapas varje gång den behövs, samtidigt som det gamla innehållet tas bort.

För att ange en mottagare i en fjärransluten dator, måste namnet föregås av systemets nodnamn och ett utropstecken. Allt som följer efter det första utropstecknet i 'mottagare' tolkas av fjärrdatorn. Om mottagare innehåller flera utropstecken kan det betyda att meddelandet ska sändas genom en sekvens av datorer på väg till den slutliga destinationen. Till exempel, om a!b!cde anges som mottagarnamn sänds meddelandet till användare b!cde på datorn a. Systemet a tolkar destinationen som en begäran att skicka meddelandet till användare cde i systemet b. Det kan vara användbart om avsändardatorn har åtkomst till system a men inte till system b, och systemet a har åtkomst till system b. *mail* använder inte *uucp* om nodnamnet anger det lokala systemnamnet (dvs lokalsystem!user).

Meddelanden till en användare lägges i användarens *mail*-fil, vanligen i biblioteket **/usr/mail**. Användarens *mail*-fil kan förändras på två sätt för att ändra funktionen hos *mail*. Filens åtkomsttillstånd för övriga kan vara läs-skriv, endast skriv eller med varken läs- eller skriv-tillstånd för olika nivåer av avskildhet. Om ändring sker till annat än standard, bevaras filen även om den är tom för att åtkomsttillstånden ska bevaras.

mail-filen kan även innehålla följande rad som första rad i filen.

```
Forward to mottagare
```

som medför att alla meddelanden till ägaren av *mail*-filen sänds vidare till 'mottagare' med tillägget "Forwarded to ..." i brevhuvudet. Detta är särskilt användbart när alla meddelanden till en person ska sändas till en maskin i en organisation med flera anslutna datorer. Installering och bort-

tagande av vidarebefodran görs med tillvalet **-F**. För att vidarebefodra all sin post till systema!användare skriver du:

```
mail -Fsystema!användare
```

För att vidarebefodra till mer än en användare skriver du:

```
mail -F"användare1,systema!användare2,systema!systemb!användare3"
```

Observera att när mer än en användare skall specificeras skall listan inneslutas av citationstecken. Listan kan vara upp till 1024 bytes lång. Antingen komma, mellanslag eller tab kan användas för att separera användare.

För att ta bort vidarebefodran skriver du:

```
mail -F ""
```

Citationstecknen krävs för att ange en tom sträng.

För att vidarebefordran ska fungera bör *mail*-filen ha *"mail"* som grupp-ID, och åtkomsttillstånden för gruppen ska vara läs-skriv.

När en användare loggar in, informeras han om det finns något meddelande. Det noteras också om ett nytt meddelande kommer in medan man använder *mail*.

TILLVAL

Tillvalen ändrar utskriften av *mail* (meddelandet).

-V	Skriver ut versionsnumret för kommandot *mail*
-e	Förhindrar utskrift av meddelanden. Utgångsstatus blir 0 om användaren har *mail*; annars blir utgångsstatus 1.
-h	Ett fönster med brevhuvuden visas istället för senaste meddelande. Detta följs av att prompten "?" visas.
-p	Alla meddelanden skrivs ut utan att prompten "?" visas.
-q	Gör att att all *mail* avbryts vid INTERRUPT (vanligen DEL). Normalt avbryts enbart eventuellt pågående utskrift av meddelanden.
-r	Gör att meddelanden visas i tidsordning, först inkommet, först utskrivet.
-fil	Gör att *mail* läser från *fil* (t ex mbox med tidigare sparade meddelanden) istället för från den normala *mail*-filen.
-Fmottagare	Lägges in i mail-filen, om den är tom, och gör så att alla inkommande meddelanden skickas vidare till *mottagare*.
-s	Hindra att en extra tom rad skapas för att markera slutet av brevhuvudet. **-s** får endast användas när en rad med Subject: finns.

-t	När man sänder mail, föregås texten av en rad som börjar med To: vilket följs av en lista med alla mottagare som detta meddelande skickades till.
-w	Sänder ett meddelande till en användare ansluten via nätverk utan att vänta på nätverksprogrammet.

FILER

/etc/passwd	identifierar avsändaren och lokaliserar mottagare.
/usr/mail/anv.namn	inkommande meddelande för användare; mail-file.
$HOME/mbox	standard för sparade meddelanden.
$MAIL	fullständiga namnet på mail-filen.
/tmp/ma*	temporär fil.
/usr/mail/*.lock	spärr för mail-biblioteket.
dead.letter	meddelande som ej kan sändas.

HÄNVISNINGAR

chmod(1), login(1), rmail(1), write(1)

ANMÄRKNING

Omständigheter kan medföra att spärrfilen inte kan tas bort. Efter avbrott kan det hända att nästa meddelande ej skrivs ut; utskrift kan tvingas fram med *mail*-kommandot **-p**.

NAMN

mesg - acceptera/förbjud kommandot *write* till egen terminalport.

SYNTAX

mesg [-V] [-n] [-y] [n] [y]

FUNKTION

Med tillvalen **-n** eller **-y** eller argumenten **n** eller **y**, ändrar kommandot *mesg* skrivtillstånden för 'alla övriga' på användarens terminalport. Detta omöjliggör eller möjliggör meddelanden sända med kommandot *write*.

Utan tillval eller argument skriver kommandot *mesg* ut den aktuella statusen som en sträng

```
is n    visas om meddelanden inte är tillåtna.
is y    visas om meddelanden är tillåtna.
```

TILLVAL

-V	Skriver ut versionsnumret för kommandot *mesg*
-y	Sätter upp skrivtillstånd.
-n	Tar bort skrivtillstånd för 'alla övriga' och avvisar inkommande meddelanden från kommandot *write*

HÄNVISNING

write(1)

FELMEDDELANDEN

Utgångsstatus från *mesg* är 0 om meddelanden kan tas emot, om inte är status 1. Vid fel är utgångsstatus 2.

MESG(1)

NAMN

mgetty - övervakning av terminaler.

SYNTAX

/etc/mgetty [-V] [-c] [-o max_öppna] [-t *sekunder*] [-f *tabfil*]

FUNKTION

mgetty startas automatiskt av *init* enligt raderna i **/etc/inittab**, för att starta övervakning av ett antal terminaler. *mgetty* används framför allt i stora system med många terminallinjer; systemets resurser behöver då inte belastas med väntande getty-processer för alla terminalportar.

Vilka terminaler som skall övervakas läses in från filen **/etc/mgettytab** eller annan angiven fil (**tabfil** nedan); terminalparametrar läses från **/etc/gettydefs**. Då bärvåg erhålls på någon av de övervakade terminalportarna startas en *getty*-process på vanligt sätt.

/etc/mgetty startas vanligen med *nice* för att vanliga terminaler skall ha lägre prioritet än huvudterminalen. Alla terminaler som övervakas av en viss */etc/mgetty* får samma prioritet.

TILLVAL

-V Visa versionen av */etc/mgetty* och avsluta sedan programmet.

-c Läs **tabfil** och skriv ut hur den tolkas av */etc/mgetty*. Om ingen tabfil anges med **-f** används standardvärdet **/etc/mgettytab**. Programmet avslutas då hela filen genomlästs.

-f tabfil Denna parameter talar om namnet på tabfilen som */etc/mgetty* läser. Om parametern inte ges till */etc/mgetty* används standardvärdet **/etc/mgettytab**.

-o max_öppna Denna parameter talar om hur många av de portar som finns i **tabfil** som skall övervakas samtidigt av *mgetty*. Detta tillval används bara för nätverksportar där det finns en hel grupp portar, men där man bara plockar upp och använder en ny av dessa i taget vid inloggning via nätverket. Kan även användas då det finns växlar.
Om parametern inte ges kommer samtliga portar i **tabfil** att övervakas.

-t sekunder Denna parameter talar om hur ofta tabfilen måste kontrolleras. Kontroll sker av ändring av filen. Om så är fallet läses den på nytt. Tiden ges i sekunder, standardvärdet är 60.

Tabfilen till */etc/mgetty* har samma utseende som **/etc/inittab**, d.v.s. formatet:

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

Fältet id: ID-kod för raden. Här kan upp till 4 tecken användas. Alla rader måste ha olika ID-koder. De används internt av */etc/mgetty*. ID-koden måste vara unik, inte bara för denna fil, utan för **/etc/inittab** och alla *mgetty*- tabfiler som används.

Fältet rstate: Här ges systemnivån: 0,...,6 eller a,b eller c. Detta fält används dock inte då */etc/mgetty* startas på en viss nivå i **/etc/inittab**.

Fältet action: Här tolkas 'once' och 'respawn' som att porten skall övervakas. Alla övriga strängar fungerar som 'off', d.v.s. respektive kommando aktiveras inte.

Fältet process: Här ges det kommando som skall exekveras då bärvåg erhålls på den port som ges. En enda typ av kommando får förekomma, nämligen *getty*. Om *nice* förekommer före *getty* kommer detta att skalas bort. Alla parametrar till *getty* kan användas, utom **-h** för hangup, som skalas bort och utförs av */etc/mgetty* själv. Om den terminalport som ges i kommandot inte finns, kommer kommandot inte att utföras.

/etc/mgetty utför följande:

- Vid systemstart läses tabfilen för vilka portar som skall övervakas.
- Vid systemstart läses också filen **/etc/gettydefs**. Detta görs för att porten skall kunna öppnas och sättas upp på ett riktigt sätt. *mgetty* gör alltid hangup på porten om inte **-h** angivits.
- Om bärvåg indikeras på en port startas en *getty* för den porten. *getty* startas med specialtillvalet **-m** för att indikera att porten redan är öppnad, och med **-h** för att hangup inte skall göras.
- Då en *getty* avslutas börjar portövervakning igen, enligt ovan.
- Status på tabfilen kontrolleras regelbundet, dels då en *getty* som varit aktiv avslutas, dels regelbundet en gång i minuten (eller kontrollerat av den tid som givits med **-t** sekunder-parametern). Om status ändrats, d.v.s. något har ändrats i filen, läses den på nytt. Om en port sätts till 'off', men är aktiv för tillfället, kommer den inte att startas om då *getty* på porten avslutas. Den kommer däremot inte att stoppas av *mgetty*.

MGETTY(1M)

EXEMPEL

Exempel på utseende av tabfilen

En tabfil till */etc/mgetty* som används för att starta fjärrinloggning kan se ut så här (filen heter **/etc/rlogintab**) :

```
r00:2:respawn:nice -16 /etc/getty -A rld00 rnet
r01:2:respawn:nice -16 /etc/getty -A rld01 rnet
...
r31:2:respawn:nice -16 /etc/getty -A rld31 rnet
```

Observera att dessa rader tagits helt oförändrade från **/etc/inittab**, där de tidigare varit placerade. Detta är meningen med att ha samma utseende av filerna **/etc/inittab** och **/etc/mgettytab**. Rader kan flyttas helt oförändrade mellan dessa. Vissa fält ignoreras dock av *mgetty*, enligt ovan.

Eftersom det står *nice* -16 före kommandona, bör den rad i **/etc/inittab** som startar *mgetty* ha ett utseende liknande:

```
mgt1:2:respawn:nice -16 /etc/mgetty -o 4 -f /etc/rlogintab
```

-o 4-tillvalet används för att bara ha 4 av de givna enheterna öppna vid varje tillfälle.

Exempel på hur utskrift av tabfilen kan kontrolleras

En tabfil till */etc/mgetty* ges nedan; filen har namnet **/etc/mterm**.

```
ut2:2:off:nice -16 /etc/getty -h -t 60 -r tty_uucp 1200
at3:2:respawn:nice -16 /etc/getty -t 60 -u 10 tty12 9600
t3:2:respawn:nice -16 /etc/getty -h -t 13 -r tty03 1200
t4:2:respawn:nice -16 /etc/getty -t 99 tty04 2400
```

En utskrift för att kontrollera tabfilen görs med:

```
$ /etc/mgetty -c -f /etc/mterm
ID ut2:  Command is IGNORED.
ID at3:
        Command='/etc/getty -t 60 -u 10 -m -h tty12 9600'
        Device='/dev/tty12'
        Speed='9600', do HANGUP
ID t3:
        Command='/etc/getty -t 13 -r -m -h tty03 1200'
        Device='/dev/tty03'
        Speed='1200'
ID t4:
        Command='/etc/getty -t 99 -m -h tty04 2400'
        Device='/dev/tty04'
        Speed='2400', do HANGUP
```

FILER

```
/etc/mgettydefs (eller angiven tabfil)
/etc/inittab
```

HÄNVISNING

getty(1M)

FELMEDDELANDEN

/etc/mgetty kan generera en del felmeddelanden. Dessa skrivs alltid på systemkonsolen.

Följande felmeddelanden skrivs på konsolen, och */etc/mgetty* avslutas:

```
mgetty: Input file '/etc/rlogintab' not found.
```

tabfilen med det givna namnet finns inte i systemet. Det är troligen felstavat, eller också har filen tagits bort.

```
mgetty: Can't open '/etc/rlogintab' for input read.
```

tabfilen med det givna namnet finns, men mgetty har inte privilegier att läsa den. Troligen har mgetty startats av en vanlig användare och inte av root.

```
mgetty: Input file '/etc/rlogintab' has no data.
```

tabfilen finns, den är läsbar, men saknar data.

```
mgetty: Can't allocate more memory.
```

Det finns inte mer minne internt för att göra 'malloc'.

```
mgetty: Can't fork to start getty.
```

Det går inte att starta en ny process. Kanske finns så många processer i systemet att den övre gränsen nåtts.

```
mgetty: EXEC of getty failed.
```

getty-kommandot som gavs i tabfilen kan **ej** startas. Kanske har ett annat program än */etc/getty* använts, och har tagits bort, eller är kanske inte exekverbart.

Ett meddelande kan ges medan *mgetty* kör. Detta skrivs också på systemkonsolen:

```
mgetty: Can't open device '/dev/rld01'.
```

mgetty har inte rättighet att öppna enheten, eller lyckas av någon annan orsak inte med detta.

NAMN

mkcfig - Ändra systemparametrar i D-NIX operativsystemfil.

SYNTAX

/etc/mkcfig [-tillval] ***osfil***

FUNKTION

mkcfig ändrar systemparametrar i D-NIX operativsystem genom att ändra i ett parameterblock i den fil operativsystemet finns. **osfil** är namnet på operativsystemfilen, normalt **dnix**.

mkcfig kan endast användas av super-user.

Kommandot *mkcfig* är interaktivt om inga tillval ges. Med tillval ändras eller skrivs systemparametrarna ut direkt. I interaktiv mod visas de aktuella parametrarna och användaren får möjligheten att ändra och skriva de nya värdena till filen.

När ändringarna är gjorda aktiveras nya parametrar bara om systemet stängs av t.ex med kommandot **/etc/shutdown -k**, och operativsystemet laddas igen.

Observera: För att nå bästa resultat krävs olika parameterar för olika system. Se D-NIX 5.3 Systemadministration för en närmare beskrivning av de olika parametrarna.

Om *mkcfig* startas utan tillval visar kommandot alla parameterar, och ändringar görs interaktivt. Texterna visas på svenska (standard är engelska) om följande parameter definieras innan start av *mkcfig*:

```
LANGUAGE=swedish
```

TILLVAL

Med tillval görs ändringarna omedelbart och de aktuella parametrarna visas inte. Värdena *n* nedan ges i decimalform.

-V	Skriver ut versionsnumret för *mkcfig*.
-a ***n***	Sätter max antal aktiva processer till *n*.
-b ***n***	Sätter filbuffertstorlek i antal 1k-block till *n*.
-c ***n***	Sätter max antal laddade filer till n.
-d ***n***	Sätter max antal semafor-justering-vid-exit till *n*.
-e ***n***	Sätter time-outtabellens storlek till *n*.
-f ***n***	Sätter max antal tillgängliga filsystem till *n*.
-g ***n***	Sätter max antal invalda minnessegment till *n*.
-k ***n***	Sätter max antal noder till *n*.
-l ***n***	Sätter max antal fillås till *n*.
-m ***n***	Sätter max antal delbara minnessegment till *n*.

-n *nnamn*	Sätter nodnamn till *nnamn*. Namnet är en sträng bestående av högst 8 tecken. Nodnamnet kan tas bort genom att ange ett minustecken inom apostrofer som namn, förutsatt att **-n** är det sista tillvalet på raden, (**-n** '-').
-o *n*	Sätter max antal öppna filer till *n*.
-p *n*	Sätter maxstorlek av "page swap area" till *n* Mbyte.
-q *n*	Sätter max antal meddelande köer till *n*.
-r	Läser konfigurationsparametrar från en rad på standard input, vilken skall ha följande format: `V=N:V=N:V=N: .....` Där: **V** är variabelnamn på en bokstav, lika som tillvalsbokstaven. **N** är decimalvärdet, eller nodnamnsträngen.
-s *n*	Sätter antalet sekunder mellan filsystemuppdateringar till *n*.
-t *n*	Sätter antalet tecken i tty inputbuffert till *n*.
-w	Skriver alla konfigurationsparametrar till standard output. Se tillvalet **-r** för formatet.
-z *n*	Sätter max antal semaforer till *n*.
-u *n*	Sätter tidsfördröjning (minuter) mellan kraftavbrottssignal och systemavstängning (**/etc/powerfail**).
-y *n*	Sätter max antal kontrollerande terminalnoder.
-j *n*	*n*=1 sätter skrivskydd (standard) varvid direkt skrivning hindras till enheter med mountade fil-system. *n*=0 tar bort skyddet.
-i *n*	Sätter säkerhetsnivån för filsystemet till *n*.
-h *n*	Sätter minnespoolen till *n* kb.

FILER

Filen med operativsystemet.

HÄNVISNING

Ingen

FELMEDDELANDEN

mkcfig ger felmeddelanden och avslutas med en utgångsstatus skild från noll om operativsystemfilen inte kan nås eller vid fel i argumentsträngen till kommandot.

NAMN

mkcont - Skapar en sammanhängande (contiguous) fil

SYNTAX

/etc/mkcont [-V] filnamn storlek

FUNKTION

mkcont skapar en fil med specificerad storlek, vilken lagras i en sammanhängande area på den fysiska enheten. En sådan fil används främst för realtidstillämpningar och databaser när snabb diskåtkomst krävs.

Filstorleken anges i bytes. Värdet kan avslutas med suffixet M, k, b eller w för att specificera multiplikation med 1024*1024 (Mbytes), 1024 (kbytes), 512 (block) eller 4 (ord). Ett par tal separerade med * indikerar en produkt.

EXEMPEL

Följande exempel skapar en fil med namnet **xfil** och storleken 1 Mbyte:

```
/etc/mkcont xfil 1M
```

ANMÄRKNING

Om det inte finns en tillräckligt stor sammanhängande area på den fysiska enheten kan *mkcont* inte skapa filen.

Om data skrivs till en sammanhängande (contiguous) fil förbi det definierade filslutet blir filens fortsättning indexerad som för en vanlig fil.

"*" har en speciell betydelse i shellen och bör omges med citations-tecken när den används tillsammans med *mkcont*.

MKCONT(1)

NAMN

mkdir - skapar nya bibliotek.

SYNTAX

mkdir [-V] [-m *mode*] [-p] bibliotek ...

FUNKTION

Kommandot *mkdir* skapar en eller flera bibliotek. För varje skapat bibliotek, lagrar systemet en post för '.' och en post för '..'. Namnet '.' är en pseudonym för biblioteket självt, och namnet '..' är en pseudonym för förälderbiblioteket. Alla bibliotek innehåller dessa poster, och endast super-user kan ta bort dem.

De skapade biblioteken får åtkomsttillstånden 777, som maskas av aktuell *umask*. Normalt blir resulterande mod 755, eftersom umask normalt är 022. Värdet umask ändras med kommandot *umask*.

Observera: *mkdir* kräver skrivrättigheter i förälderbiblioteket för så vitt inte kommandot utförs av super-user.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *mkdir*
-m*mode*	Sätter åtkomsträttigheterna på det/de skapade biblioteken till *mode*
-p	*mkdir* skapar alla icke-existerande fadersbibliotek först

FILER

Inga

EXEMPEL

```
mkdir Progex
```

Skapar ett nytt underbibliotek till det aktuella biblioteket. Om detta inte är möjligt, visas ett felmeddelande. Prompten skrivs ut när kommandot är klart. (För att se det skapade biblioteket, skriv *ls*-**d** Progex).

```
mkdir -p a/b/c
```

Skapar biblioteket c men även a och b om de ej finns.

```
mkdir -m 700 bib
```

Skapar biblioteket bib med åtkomsträttigheterna 700 om ej *umask* är satt till ett värde som påverkar detta.

HÄNVISNING

rm(1), rmdir(1), sh(1), umask(1)

FELMEDDELANDEN

mkdir ger en utgångsstatus skild från noll och skriver ut ett felmeddelande om ett bibliotek inte kan skapas. I annat fall ges 0 som utgångsstatus.

MKDIR(1)

NAMN

mkfs - Skapa ett D-NIX filsystem

SYNTAX

/etc/mkfs [-V] [tillval] enhetsfil

FUNKTION

Kommandot *mkfs* skapar ett filsystem på enheten eller i filen enhetsfil enligt specificerade tillval. Ett filsystem måste skapas på en enhet, innan det kan läggas till med kommandot *mount*.

Kommandot *mkfs* testar volymen innan filsystemet skapas. Alla fysiska enheter, där ett filsystem ska skapas, måste först vara formaterade.

Om inga tillval är givna och om enhetsfilen är en fysisk enhet, använder *mkfs* standardparametrar och testar volymstorleken genom att söka slutet på volymen.

mkfs skapar alltid följande systemfiler i det nya filsystemet. Användadet av dessa filer är begränsat.

/lost+found
bibliotek som används av kommandot *fsck*
/..systemfiles
bibliotek med följande innehåll:
bitmap: bitmap över volymen
badspots: en fil för felaktiga eller oanvända block
inodelist: fil för volymens inodlista.
swaparea: system swaparea (om specificerad)
volumedescr: volymbeskrivningsfil

TILLVAL

Alla numeriska värden i tillvalen kan följas av k, K, m or M för att indikera multiplikation med 1024 eller 1024*1024.

-V	Skriver ut versionsnumret för kommandot *mkfs*
-adress *filnamn*	Kopierar ett *filnamn* till angiven adress i det nya filsystemet. Adressen anges i decimalform. Utrymme reserveras i bitmappen. Tillvalet är användbart i BOOT-procedurer.
-b *bstorlek*	Sätter blockstorleken för volymen till *bstorlek*. *bstorlek* måste vara en multipel av 2 i området 256 - 16384. Standard blockstorlek är 2 kbyte.
-n *nmax*	Allokerar utrymme för *nmax* inoder på volymen. Standard är 1/32 av volymstorleken. Detta segment garanteras vara kontinuerligt. Om fler inoder behövs, utökas inodlistan automatiskt på samma sätt som för en ordinarie fil. Maximalt 65535 inoder kan allokeras.
-r *bibliotek*	*bibliotek* kopieras till rootbiblioteket i det nya filsystemet. Alla tillhörande filer och bibliotek kopieras rekursivt. Filerna kopieras med sina ursprungliga status

och modifieringstider. Det gäller alla filer utom filer i /dev, vilka innehåller enhetsparametrar till värd-datorn. Den nya enhetens status läggs till när kopieringen är gjord. Filen 'enhetsfil' får inte vara en del av hierarkin som kopieras.

-s ***sstorlek*** Allokerar en kontinuerlig "swap area" på volymen med storleken *sstorlek* bytes och namnet **/..systemfiles/swaparea**. Detta tillval används inte nu längre.

-v ***vstorlek*** Sätter volymstorleken till det antal block som anges av *vstorlek*. Som standard på en fysisk enhet, utan -v eller -t tillval, känner *mkfs* av den fysiska storleken och använder detta som volymstorlek. När ett filsystem skapas i en fil, måste däremot alltid volymstorleken anges.

-t ***vstorlek*** Alternativt sätt att ange volymstorlek. *vstorlek* anges normalt i bytes. Tecknen k eller M, som sista tecken i *vstorlek*, anges att storleken är i kbytes eller Mbytes. *mkfs* känner av den fysiska storleken av enheten då tillvalen -v eller -t ej anges, och använder då detta som volymstorlek. När ett filsystem skapas i en fil, måste däremot volymstorleken alltid anges.

-x ***xfil*** Läser definitionstabellen för badspots från *xfil*. Varje badspot beskrivs som en post med formen m,n där m är startblock och n är längden (i block) på varje dålig area i enheten. Postens separeras med new-line.

FILER

/lost+found
/..systemfiles/badspots
/..systemfiles/bitmap
/..systemfiles/inodelist
/..systemfiles/swaparea (om specificerad)
/..systemfiles/volumedescr

EXEMPEL

Exempel 1:

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

Detta exempel skapar ett filsystem på en redan formaterad 5 1/4 tums diskett och använder 2kbytes blockstorlek (standard) efter att ha känt av volymstorleken genom att läsa disketten.

Exempel 2:

```
/etc/mkfs -b 1024 -t 720k /dev/mf0
```

Här är blockstorleken satt till 1 kbyte och volymstorleken är direktangiven till 720 kbytes i ett nytt filsystem på en 5 1/4 tums diskett.

HÄNVISNING

fsck(1M), mount(1M), umount(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

Volymstorleken måste specificeras med antingen **-v** eller **-t** tillvalet om enhetsfilen är en fil och inte en fysisk enhet.

MKFS(1)

NAMN

mknod - skapar specialfiler eller pipes.

SYNTAX

/etc/mknod [-V] namn [c | b] primär sekundär

/etc/mknod [-V] namn p

FUNKTION

mknod skapar specialfiler i ett bibliotek och i motsvarande inod. Specialfilen är antingen en ny fysisk enhet eller en namngiven pipe-fil.

Endast super-user har lov att skapa fysiska enheter, men vilken användare som helst kan skapa sin egen pipe-fil.

Namn är en komplett sökväg (pathname) för den nya enheten eller den namngivna pipe-filen och argumenten c, b eller p bestämmer typen av enhet eller fil. Fysiska enheter skall alltid finnas i biblioteket **/dev**.

c	Detta är den normala enhetstypen för fysiska enheter. Överföring till/från enheten sker med den fysiska blockstorleken. För t ex terminaler och skrivare sker överföringen teckenvis.
b	Överföringen kan ske med valfri blockstorlek genom att buffring alltid sker i systemet.
p	Denna form av kommandot *mknod* skapar en namngiven pipe-fil, dvs. en specialfil i filsystemet, vilken man läser och skriver som en FIFO-buffer, liksom med en pipe i *sh*. Filen kan tas bort med kommandot *rm*.
primär	Primär enhetstyp (major device), anger vilken systemrutin som skall användas för åtkomst till den skapade enheten. Den primära enhetstypen anges oktalt om första siffran är 0, i annat fall decimalt.
sekundär	Sekundär enhetstyp minor device anger en parameter till systemrutinen t ex, ett fysiskt enhetsnummer. Den sekundära enhetstypen anges oktalt om första siffran är 0, i annat fall decimalt.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *mknod*.

FILER

/dev/*

EXEMPEL

Exempel 1:

```
/etc/mknod      $HOME/PIPEFILE p
```

Skapar en namngiven pipe-fil i användarens hembibliotek. Skrivning till denna filen görs alltid genom tillägg i slutet av filen. Läsning sker alltid från början och varje läst tecken raderas automatiskt från filen efter läsningen.

HÄNVISNING

rm(1), sh(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Då en ny enhet skapats, kan åtkomsttillstånden behöva ändras vilket görs med kommandot *chmod*.

Definiering av primära och sekundära enhetsnummer är unika för varje system. Se **Systemadministration**.

Fysiska enheter så väl som namngivna pipes tas bort med kommandot *rm* efter filnamnet.

NAMN

mknodm - skapar specialfiler för spegel-skivminnen

SYNTAX

/etc/mknodm [-V] namn [c | b] prim1 sek1 prim2 sek2

FUNKTION

mknodm skapar en specialfil i ett bibliotek och motsvarande inod. Specialfilen är en ny fysisk enhet för ett spegelskivminne (mirror-disk).

Endast super-user har lov att skapa fysiska enheter.

Namn är en komplett sökväg (pathname) för den nya enheten och argumenten c eller b bestämmer typen av enhet. Fysiska enheter skall alltid finnas i biblioteket **/dev**.

Enhetstypen för skivminnen är normalt 'c' i D-NIX.

Parametrarna prim1, sek1 för det ena fysiska skivminnet och prim2, sek2 för det andra, skall ange olika fysiska skivminnen. Se **Systemadministration** samt dokumentationen för 'mirror-disk' för detaljer.

c	Detta är den normala enhetstypen för fysiska enheter. Överföring till/från enheten sker med den fysiska blockstorleken. För t ex terminaler och skrivare sker överföringen teckenvis.
b	Överföringen kan ske med valfri blockstorlek genom att buffring alltid sker i systemet.
prim1, prim2	Primära enhetsnummer "major devices", anger vilka systemrutiner som skall användas för åtkomst till de fysiska enheterna. Det primära enhetsnumret anges oktalt om första siffran är 0, i annat fall decimalt.
sek1, sek2	Sekundära enhetsnummer "minor devices" anger parametrar till systemrutinen t ex, fysiska portar. De sekundära enhetsnumren anges oktalt om första siffran är 0, i annat fall decimalt.

TILLVAL

-V	Skriver ut versionsnumret för kommandot ***mknodm***.

FILER

/dev/*

EXEMPEL

```
/etc/mknodm /dev/simirr c 5 18 5 2
```

Skapar enheten **/dev/simirr** som en 'mirror-disk' där huvudskivminnet (vid *mount*) har sekundära enhetstypen (minor) 18 och spegel-skivminnet har sekundära enhetstypen (minor) 2.

Notera att detta endast är ett exempel. Vilka sekundära enhetstyper som skall användas beror både på datortypen och vilka skivminnen som används.

HÄNVISNING

rm(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Då en ny enhet skapats, kan åtkomsttillstånden behöva ändras vilket görs med kommandot *chmod*. Skivminnen skall av säkerhetsskäl alltid vara skyddade mot skrivning och läsning för alla utom super-user (root).

Definiering av primära och sekundära enhetsnummer är unika för varje system. Se **Systemadministration**.

Fysiska enheter kan tas bort med kommandot *rm*.

NAME

mksort - skapa en sorteringsfil för kommandot *sort*

SYNTAX

/etc/mksort [-V] [-f] ***sortfil***

FUNKTION

Kommandot skapar en ny binär sorteringstabell med 256 bytes. Sorteringstabellfilen definierar ordningsföljden i kommandot *sort*. I filen ges 256 tecken i ordningsföljd efter sorteringsvärden.

Med tillvalet **-f** skapas istället motsvarande tabellfil för konvertering mellan versaler och gemener. Då får *sortfil* automatiskt tillägget .f efter filnamnet.

Kommandot *sort*, givet utan tillvalet **-s**, antar att namnet på den binära sorteringstabellfilen är **/usr/etc/sorttable** och på filen med konverteringstabellen **/usr/etc/sorttable.f**.

mksort visar varje tecken (som standard antages ASCII-tecknet med värdet lika med ordningsnumret) och användaren svarar med RETUR enbart eller med ett annat tecken som skall ersätta standardtecknet. Exempelvis kan en tabell med korrekt sortering av svenska Å Ä Ö göras genom att ange Å, Ä, Ö samt å, ä, ö som tecken nummer 93, 91, 92 samt 125, 123, 124.

Kontrolltecken visas med ^ som prefix. Tecken med 8:e biten satt visas med prefix ~.

TILLVAL

-V	Skriver ut versionsnumret för kommandot mksort.
-f	Skapar en tabellfil (med tillägget .f till filnamnet) för jämförelser (*sort* **-f**), där versaler och gemener anses lika.

FILER

/usr/etc/sorttable Standardfil för sorteringsordningen.
/usr/etc/sorttable.f Standard tabellfil för konvertering gemena/versala.

HÄNVISNING

sort(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Kontrolltecken och tecken med 8:e biten satta kan för närvarande inte matas in.

MKSORT(1)

NAMN

mkuser - lägger till användare

SYNTAX

/etc/mkuser [-V] [användarnamn]

FUNKTION

mkuser lägger till en ny användare genom att modifiera filerna **/etc/passwd** och **/etc/group** för att skapa ett hembibliotek och en **.profile** fil för den nya användaren.

Observera: Kommandot kan endast exekveras av en super-user.

Kommandot innehåller ett antal frågor vilka finns beskrivna nedan. Standardvärden visas inom parentes då frågorna ställs.

User name: Namnet på den nya användaren. Om detta redan är använt ställs frågan om på nytt och man får ge ett nytt användarnamn. Slår man bara RETUR avbryts kommandot. Namnet kan anges direkt på kommandoraden, varvid denna fråga hoppas över.

User-id: Den nya användarens ID-nummer. Detta måste vara ett heltal större eller lika med 1. Det rekommenderas att använda värden över 100 för vanliga användare. Om den givna identi- fikationen redan är använd kommer frågan att upprepas. Svarar man genom att trycka på return fås första lediga ID-nummer över 100.

Password: Lösenord som skall användas vid inloggning. Endast RETUR gör att enbart RETUR används som lösenord.

Group-name: Namnet på den grupp som användaren skall tillhöra. Genom att enbart trycka på return så antas gruppnamnet other. För att lista alla användargupper, svara med ett ?.

Group-id: Användarens grupp-idnummer. Funktionen är densamma som vid inmatning av användar-id. Om enbart return har slagits vid föregående fråga (grupp other valdes) eller om det valda gruppnamnet redan finns i filen **/etc/group**, kommer denna fråga att hoppas över och grupp-id visas som det är definierat i denna fil. Endast return vid nytt grupp-ID, kommer att ge första lediga grupp-ID.

Password: Lösenord som skall användas vid byte av grupptillhörighet. Slå endast RETUR om inget lösenord behöver anges. Om gruppnamnet redan finns i filen **/etc/group** hoppas denna fråga över.

Extra user-description:

Godtycklig sträng som dock inte får innehålla kolon (:). Vissa reserverade textsträngar för kommandot *login* kan anges här. Se *login*.

Login-directory: Användarens hembibliotek. Standard är **usr/ användarnamn**. Komplett namn måste anges t.ex **/usr/olle.** Om biblioteket redan finns, får man verifiera sitt svar. Om man då endast anger RETUR kommer kommandot *mkuser* att avbrytas.

Start-program: Det program som skall startas direkt vid inloggning. Om man endast trycker RETUR kommer */bin/sh* att användas. Om det givna programmet inte existerar måste svaret verifieras. **Observera:** Ett komplett programnamn måste anges.

Terminal-type (VT100):

Den terminaltyp som användaren förväntas arbeta vid. I filen **.profile**, vilken skapas av *mkuser* , kommer variabeln TERM att sättas till angivet värde. Om enbart RETUR anges sätts den till standardvärdet inom parenteserna.

TILLVAL

-V Skriver ut versionsnumret för kommandot *mkuser*

FILER

/etc/group
/etc/passwd

EXEMPEL

Exempel 1:

```
/etc/mkuser
```

Alla ovanstående frågor måste besvaras.

Exempel 2:

```
/etc/mkuser olle
```

Lägg till en användare med namnet olle. Alla frågor utom den första måste besvaras.

HÄNVISNING

rmuser(1M), sh(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Undvik tecknen '#' och '@', eftersom dessa speciellt avkodas av processen *getty* vid inloggning. Se kommandot *getty*.

MKUSER(1M)

NAMN

mntchk - kontrollerar en enhet och lägger till den (*mount*)

SYNTAX

/etc/mntchk [-V] [filhanterare] enhet bibliotek

FUNKTION

mntchk kontrollerar med hjälp av kommandot *fscl* om filsystemet på den specificerade enheten är rent. Om så inte är fallet, utförs kommandot *fsck*. Därefter utförs kommandot *mount*.

Kommandot *mntchk* skriver ut meddelanden om vad det gör på standard output.

Det valfria argumentet filhanterare anger vilken filhanterare som skall användas för åtkomst till filer i filsystemet. Argumentet enhet är den enhet (eller fil) där filsystemet finns och argumentet bibliotek talar om till vilket bibliotek filsystemet skall anslutas.

Se kommandot *mount* för ytterligare information.

Observera: Avskilj alltid filsystemet med kommandot *umount* innan enheten tas bort, t.ex innan en diskett med ett filsystem tas ut.

TILLVAL

-V Skriver ut versionsnumret för kommandot *mntchk*.

FILER

/etc/mnttab
/etc/mtab

EXEMPEL

Exempel 1:

```
/etc/mntchk /dev/mf0 /mnt
```

Detta kommandot lägger till ett filsystem med D-NIX standard filhanterare i enheten **/dev/mf0** via biblioteket **/mnt.**

HÄNVISNING

mount(1M), umount(1M), fscl(1M), fsck(1M)

FELMEDDELANDEN

Felmeddelanden ges och utgångsstatus blir skild från noll om kommandot misslyckas.

MNTCHK(1M)

NAMN

mount, umount - lägga till eller ta bort filsystem.

SYNTAX

/etc/mount [-V] enhet bibliotek [-r]

/etc/mount [-V] filhanterare enhet bibliotek [parametrar]

/etc/umount enhet/bibliotek

FUNKTION

Om inget argument är angivet, listar *mount* alla externa enheter som är mountade i systemet.

Kommandot *mount* informerar kärnan i operativsystemet om förekomsten av filsystem som kopplas till det befintliga filsystemet. Det första kommandoformatet ovan används för standard filsystem, medan det andra formatet används för tilläggning av icke-D-NIX filsystem genom användande av särskild filhanterare.

Kommandot *umount* informerar kärnan i D-NIX om att det tidigare mountade, filsystemet kopplas bort. Alla befintliga systembuffrar rensas och länken till det resterande filsystemet tas bort. Detta är nödvändigt innan någon extern enhet flyttas fysiskt, t.ex en diskett. Ett filsystem kan inte kopplas bort om någon fil på det är öppen eller om ett bibliotek i det är i bruk av någon användare.

Argumentet enhet anger namnet på den externa enhet (specialfil i biblioteket **/dev**) innehållande det filsystem som skall mountas. Om särskild filhanterare används kan argumentet enhet också specificera en fil i det aktuella systemet, som då skall innehålla kopia av ett komplett filsystem.

Argumentet bibliotek är namnet på det biblioteket som skall innehålla rooten till det nya filsystemet. Argumentet måste vara ett existerande och tomt bibliotek. Om enhet specificerar en fil som har en kopia av filsystemet, används inte argumentet bibliotek, det måste dock anges på kommandoraden. Rooten till det nya filsystemet finns då i den angivna filen (enhet).

Observera: Använd kommandot *umount* för att koppla bort ett mountat filsystem, innan någon slags fysisk bortkoppling sker i eller av den externa enheten, t.ex att en diskett tas ur.

I den andra formen av kommandot *mount*, ges argumentet filhanterare för att göra det möjligt för en icke-D-NIX filhanterare att ta hand om det mountade filsystemet. Filhanterare finns normalt i systembiblioteket **/hnd**. På detta sätt har operativsystemet direktåtkomst till andra typer av filsystem, antingen på diskett eller i en fil på systemets winchesterskiva. Mountas ett filsystem via en filhanterare startas styrprogrammet och alla argumenten till kommandot *mount* överförs till filhanteringsprogrammet. Specialparametrar kan användas, beroende på hanterarens behov.

Endast super-user kan använda kommandona *mount* och *umount*.

TILLVAL

-V	Skriver ut versionsnummer av kommandot *mount*
-r	Lägger till filsystemet med endast läs-rättigheter. Filsystem som ligger på kassetter, eller på skrivskyddade fysiska enheter, måste läggas till på detta sätt. I annat fall kommer ett felmeddelande att ges när försök görs att förändra åtkomsttider. Detta gäller även om inget skrivs till filer i filsystemet.

FILER

/etc/mnttab	'mount'-lista till D-NIX filsystem.
/etc/mtab	'mount'-lista till filsystem som är inkopplade med en speciell filhanterare.
/mnt	Ofta använt standardbibliotek för tillägg av externa filsystem.
/hnd	Bibliotek med speciella filhanterare.
/tmp/mtab_lock	Temporär låsfil, som hindrar att flera mount körs samtidigt.

EXEMPEL

Exempel 1:

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

Filsystemet på enheten **/dev/mf0** (5 1/4 tums diskettenhet) läggs till och nås via biblioteket **/mf0**. Detta biblioteket skall finnas i systemet och måste vara tomt. **Observera:** Innan disketten tas ut måste kommandot *umount* användas enligt nedan:

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

Exempel 2:

```
/etc/mount /dev/mf0 /usr/kkp/rst -r
```

Filsystemet på enheten **/dev/mf0** läggs till och nås via biblioteket **/usr/kkp/rst**. Det nya filsystemet kan endast läsas. **Observera:** Innan disketten tas ut måste kommandot *umount* användas enligt ovan.

Exempel 3:

```
/etc/mount /hnd/msdosfh /dev/mf0 /mnt
```

Ett filsystem som inte är D-NIX kompatibelt läggs till med den speciella filhanteraren */hnd/msdosfh*. Filsystemet finns på en 5 1/4 tums diskett och nås genom biblioteket **/mnt**.

Observera: Innan disketten tas ut måste kommandot *umount* användas enligt ovan. Främmande fil-hanterare är inte inkluderade i D-NIX standard mjukvara.

HÄNVISNING

mntchk(1M), umount(1M), fscl(1M)

FELMEDDELANDEN

Följande utgångsstatus returneras efter exekvering av komandona *mount*/*umount*.

0 *mount*/*umount* lyckades.
1 *mount*/*umount* lyckades inte.
2 Försök att lägga till en struktur som inte är ett rent filsystem.

mount/*umount* ger följande felmeddelande om någon annan samtidigt använder kommandot, dvs om filen **/tmp/mtab_lock** redan finns.

```
ERROR: Can't get exclusive access to mount table
```

ANMÄRKNING

Ett orent filsystem kan inte läggas till. Se *fscl* eller *fsck*.

Filen **/etc/mtab** töms alltid och filen **/etc/mnttab** initieras när systemet startas.

Som standard blir alla mountade filsystem skyddade mot skrivning i rå mod. Jämför kommandot */etc*/*mkcfig*.

NAMN

mv - flyttar och/eller byter namn på filer och bibliotek.

SYNTAX

mv [-V] [-f] filnamn1 filnamn2

mv [-V] filnamn .... bibliotek

mv [-V] bibliotek1 bibliotek2

FUNKTION

Kommandot *mv* används antingen för att ändra namn på filer eller flytta filer till ett annat bibliotek. Att flytta filer innebär oftast bara att filreferenserna (länkarna) i de berörda biblioteken ändras. Om däremot filerna flyttas till ett annat filsystem, görs en kopia av dem. Jämför kommandot *cp*, som kopierar filer utan att ta bort originalen.

mv kan också byta namn eller flytta ett bibliotek, dock bara inom samma filsystem.

Se även kommandot */etc/mvdir*, som endast kan flytta och/eller byta namn på bibliotek inom ett filsystem. Det kräver super-user rättigheter för att kunna användas.

Första syntaxen ovan används för att byta namn på en fil. Om båda filerna som anges befinner sig i samma bibliotek kommer filnamn1 att byta namn till filnamn2. Om filerna befinner sig i olika bibliotek flyttas filnamn1 till filnamn2's bibliotek. Om filnamn2 redan existerar, kommer i båda dessa fall innehållet i filnamn2 att förstöras innan namnet filnamn2 ges till filen filnamn1.

Om åtkomsttillstånden för filen filnamn2 inte tillåter skrivning, stoppas kommandot och filnamn2 och dess åtkomsttillstånd skrivs ut på standard output. Systemet väntar därefter på ett svar från standard input, innan det kan fortsätta. Exekveringen fortsätter om svaret börjar med y och användaren har tillstånd att ta bort filnamn2. Om inte, avbryts kommandot. Om tillvalet **-f** ges eller om standard input inte är en terminal, ställs ingen fråga och flyttningen genomförs, om möjligt.

När ett befintligt bibliotek ges som sista argument (andra syntaxen ovan), flyttas alla angivna filer till detta bibliotek.

Om ett bibliotek flyttas till ett annat bibliotek (sista två syntaxexemplen ovan), beror resultatet på om bibliotek2 finns eller inte. Om det inte gör det, flyttas bibliotek1 och får namnet bibliotek2. Om bibliotek2 finns, flyttas bibliotek1 och blir ett underbibliotek till bibliotek2. Observera att bibliotek2 inte kan vara ett underbibliotek till bibliotek1.

Det är inte möjligt att flytta en fil till sig själv med detta kommandot.

TILLVAL

-V	Skriver ut versionsnumret till kommandot *mv*
-f	Genomför flyttningen utan frågor, om det är tillåtet, även om inget skrivtillstånd finns till filen filnamn2.

FILER

/usr/lib/mv_dir

EXEMPEL

Exempel 1:

```
mv temp temp2
```

Om filen temp2 inte existerar, skapas den. Om den finns, förstörs innehållet i den. Därefter flyttas innehållet i temp till temp2. Sedan förstörs temp. Kontrollera resultatet med kommandot *ls*. En flyttning betyder normalt inte att innehållet kopieras, endast att filnamnet ändras i biblioteken. Observera dock anmärkningen nedan.

Exempel 2:

```
mv Kap1 Kap2 Kap3 Kap4 /manual
```

Filerna Kap1 - 4 flyttas till biblioteket /manual. Alla fyra filerna kopieras med sina ursprungsnamn och orginalen raderas.

Exempel 3:

```
mv mydir /usr/com/yourdir
```

Biblioteket **mydir**, i det aktuella biblioteket, flyttas och blir underbibliotek till det redan befintliga **/usr/com/yourdir**. Alla filer och underbibliotek i **mydir** flyttas tillsammans för att bibehålla trädstrukturen. Det nya biblioteket kommer att få namnet **/usr/com/yourdir/mydir**. Användaren måste ha skriv- och exekveringsprivilegier i det andra biblioteket.

Exempel 4:

```
mv /usr/me/mydir /usr/com/yourdir
```

Om biblioteket **/usr/com/yourdir** inte finns, flyttas och namnändras **/usr/me/mydir** till **/usr/com/yourdir**.

HÄNVISNING

cp(1), copy(1), chmod(1), mvdir(1M), rm(1)

ANMÄRKNING

Om filnamn1 och filnamn2 tillhör två olika filsystem, kopierar *mv* även innehållet i filen och tar sedan bort originalet. I detta fall ändras ägarnamnet till namnet på den användare som exekverat kommandot *mv* och alla länkar till andra filer förloras.

Innehållet i ett befintligt bibliotek kan aldrig skrivas över när ett bibliotek flyttas med kommandot *mv*, filer kan dock skrivas över.

Pathnamn som innehåller .. (förälderbiblioteket) är inte tillåtna vid flyttning av bibliotek.

NAMN

mvdir - flyttar ett bibliotek

SYNTAX

/etc/mvdir [-V] dirname newname

FUNKTION

mvdir flyttar bibliotek inom ett filsystem. dirname måste vara ett bibliotek. newname får inte vara ett underbibliotek till dirname. Om newname finns, måste det vara ett självständigt bibliotek och dirname flyttas och blir ett underbibliotek till newname. Om newname inte finns, flyttas dirname och får namnet newname. Endast super-user kan använda *mvdir*.

Se också kommandot *mv*, vilket också kan användas för att flytta bibliotek.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *mvdir*

FILER

/usr/lib/mv_dir

EXEMPEL

Exempel 1:

```
/etc/mvdir /usr/alpha/mittbib /usr/beta/nyttbib
```

Detta kommandot, som ges av super-user, flyttar alla filer och underbibliotek i **/usr/alpha/mittbib** till det nya biblioteket **/usr/beta/nyttbib**.

Exempel 2:

```
/etc/mvdir ornew ornewbib
```

Biblioteket **ornew** får istället namnet **ornewbib**, förutsatt att **ornewbib** inte existerar. Om inte, flyttas **ornew** och blir ett underbibliotek till **ornewbib**, dvs. det får namnet **ornewbib/ornew**. *mv* kan även användas.

HÄNVISNING

mkdir(1), mv(1)

MVDIR(1M)

NAMN

newgrp - logga in till en ny grupp

SYNTAX

newgrp [-V] [-] grupp

FUNKTION

Kommandot *newgrp* ändrar användarens grupp-ID. Användaren förblir inloggad i det aktuella biblioteket men åtkomsträttigheter till filerna bestäms av det nya verkliga och gällande grupp-ID. Aktuell shell ersättes med en ny shell vid exekvering av kommandot *newgrp*, vare sig gruppändringen accepteras eller ej.

Kommandot frågar efter grupp-lösenord, om det är specificerat i filen **/etc/group**, men endast om användaren saknar eget password eller om han inte är listad som grupp-medlem i **/etc/group**.

Kommandot *newgrp* utan argument ändrar grupp-ID tillbaka till den grupp som finns angiven i **/etc/passwd** för användaren ifråga.

Medan shell ersätts behåller exporterade variabler sina värden. Systemvariabler (t ex PS1, PATH m fl) så väl som icke-exporterade variabler 0-ställs eller återställs till sina standardvärden (Se *export* och *sh*).

Om det första argumentet är -, sätts miljön upp som om användaren loggat in igen enligt specifikationer i filen **/etc/passwd** (utom grupp-ID) och i filerna **/etc/profile** och **.profile**.

Kommandot *newgrp* gör att nuvarande shell byts ut mot en ny shell utan att skapa en ny process. Omdirigering av indata/ utdata bör undvikas då den nya shell startas i interaktiv mod.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *newgrp*

FILER

/etc/group
/etc/passwd

HÄNVISNING

login(1), passwd(1), sh(1), export(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Grupplösenordet i filen **/etc/group** sätts upp med kommandot:

```
passwd -f/etc/group
```

NEWGRP(1)

NICE(1), NOHUP(1)

NAMN

nice - utför ett kommando med lägre prioritet.

nohup - utför ett kommando med spärr mot hangup och avbrottssignaler.

SYNTAX

nice [-V] [-nummer] kommando [argument]

nohup [-V] kommando [argument]

FUNKTION

Kommandot *nice* används till att sänka prioriteten på kommandon och är lämpligt att använda då kommandon ska utföras i bakgrunden. Kommandon som startas med *nice* tar ofta längre tid men försenar inte andra normala processer. Prioriteten ändras genom att ett värde anges i argumentet -nummer. Prioriteten räknas ned med det nummer som anges, dvs desto högre nummer desto lägre prioritet. Standardvärdet är -10, maxvärdet är -19. Högre värden sätts ner till -19.

Super-user kan utföra kommandon med högre prioritet än normalt genom att ange negativ prioritet t ex --4.

Processer i D-NIX kan vara antingen realtidsprocesser eller normala processer med time sharing (tidsdelning). Med *nice* påverkas den time sharing-prioritet som ett kommando startas med i området 4 .. 39.

Grundprioriteten i systemet är prioritet 4, men standard är dock 20 för kommandon från en terminal, enligt filen **/etc/inittab**. Kommandon från systemets huvudterminal har högre prioritet (4), men vid inloggning som vanlig användare (ej super-user) sänks ändå prioriteten automatiskt till 20.

Kommandot *nohup* förhindrar hangup (*kill* 1) och uthoppssignalen (*kill* 3) från att påverka kommandot. Detta möjliggör fortsatt exekvering av ett kommando efter utloggning, även om kommandot självt inte kan hantera dessa signaler. För att förhindra även avbrottssignalen (*kill* 2) bör kommandot ges som bakgrundsprocess (se nedan). Om inte standard output och standard error output är uttryckligen omdirigerade tvingar *nohup* alla utdata till filen **nohup.out**. Utan skrivrättighet i det aktuella biblioteket placeras filen **nohup.out** i användarens hembibliotek **$HOME/nohup.out**.

För att starta ett kommando i bakgrunden sätts tecknen & som den sista parametern på kommandoraden. Bakgrundskommandon påverkas inte av avbrottssignalen (*kill* 2).

TILLVAL

-V	Skriver ut versionsnummer för *nice* och *nohup*

FILER

/etc/inittab	Anger standard prioritet.
nohup.out eller $HOME/nohup.out	standardfil för utdata med kommandot *nohup*.

EXEMPEL

Exempel 1:

```
nice -4 copy utvprog /vprog/tprog
```

Kommandot *copy* utförs med prioritet 24, med förutsättningen att aktuella prioriteten för shell är 20. Detta möjliggör att andra processer utförs under tiden med minsta dröjsmål.

Exempel 2:

```
nice sh progman &
```

Kommandot anropar ett shell med lägre prioritet för bearbetning i bakgrunden av kommandon i kommandofilen **progman**. Prioriteten sänks med 10 enligt standard.

Exempel 3:

```
nohup sh progman &
```

Kommandot anropar ett shell som inte ska påverkas av signaler för hangup eller uthopp och som bearbetar kommandona i kommandofilen **progman** i bakgrunden. Användaren kan logga ut ur systemet men bearbetningen av kommandona kommer att fortsätta utföras tills de är klara. Om standard output eller standard error output inte är ändrat i progman styrs all utdata till **nohup.out**.

Exempel 4:

```
nice nohup progman &
```

Detta är samma exempel som 3, men denna gång förutsättes att kommandofilen har exekverings-tillstånd och kommandon exekveras med lägre prioritet genom att använda standardvärdet -10.

HÄNVISNING

sh(1), kill(1), ps(1)

FELMEDDELANDEN

nice returnerar utgångsstatus för det utförda kommandot.

ANMÄRKNINGAR

Bara ett kommando kan ges efter *nohup*. Flera kommandon inom parentes eller pipelines är inte tillåtna. Enda sättet att utföra flera kommandon eller pipelines är att inkludera dessa i en kommandofil som startas med *nohup*.

NAMN

od - oktal dump.

SYNTAX

od [-Vabcdhosx] [filnamn] [+]offset[.][b]]

od -V -a -b -c -d -h -o -s -x filnamn +offset.b

FUNKTION

Kommandot *od* producerar en listning av den specificerade filen, eller av standard input om ingen fil har angivits. Filen kan listas i flera olika format, oktala ord (standard), oktala bytes, ASCII-tecken, hexadecimala eller decimala ord. De olika formaten kan produceras tillsammans eller separat. Normalt listas filen med start från början om offset inte har angetts. Listan är till standard output.

Som första argument anges vilken fil som ska listas. Om inget filnamn anges används 'standard input'.

Offset som kan anges som andra argument används för att kontrollera avståndet från början av filen till den position där listningen skall börja. Om inget filnamn är specificerat måste offset föregås av tecknet +. Normalt tolkas offset som antal bytes oktalt. Läggs tecknet '.' till offset tolkas det som decimala bytes. Om **b** läggs till efter offset tolkas det som antal 512 bytes block. Utan tecknet '.' tolkas offset som hexadecimalt om det börjar med 0x eller 0X.

Listningen fortsätter till filens slut eller tills kommandot avbryts.

TILLVAL

-V Skriver ut versionsnumret för kommandot *od*

-a Skriver ut även flera lika utdata-rader i listningen. Som standard skriver *od* en stjärna (*) på en annars tom rad för att visa att en eller flera rader av listningen, likadana som den föregående inte skrivits ut.

-b Skriver ut innehållet byte-vis, som standard oktalt om inte något annat tillval ges samtidigt.

-c Skriver ut innehållet som ASCII-tecken. Vissa icke skrivbara tecken visas enligt nedan. Alla andra icke skrivbara tecken representeras av sina byte-värden.

```
\0 null (ascii-värdet 0)
\b backspace
\f formfeed (ny sida)
\n newline (ny rad)
\r return
\t tab
```

-d Skriver ut innehållet i decimala 16-bitars ord, utan att stora tal tolkas som negativa.

-h Skriver innehållet som hexadecimala 16-bitars ord.

-o	Skriver ut innehållet i oktala 16-bitars ord. Standard om inget tillval anges.
-s	Skriver ut innehållet i decimala 16-bitars ord med tecken, dvs stora tal skrivs ut som negativa.
-x	Skriver innehållet som hexadecimala 16-bitars ord.

Om **-d, -o, -s, -h** eller **-x** kombineras med **-b** eller **-c** skrivs varje byte ut i respektive format.

FILER

Inga

EXEMPEL

Exempel 1:

```
od -c Kap1
```

Filen Kap1 listas som ASCII-tecken.

Exempel 2:

```
od /dev/mf0 +1024.
```

Disketten som specificerats som **/dev/mf0** läses med början vid 1024 bytes från diskettens början.

Exempel 3:

```
od /dev/mf0 +2.b
```

En diskett i enheten **/dev/mf0** läses med början vid det tredje blocket, dvs. 2x512 = 1024 bytes från början av disketten.

Exempel 4:

```
od -xb myfile
```

Innehållet i **myfile** listas på standard output. **-xb** anger listning i hexadecimala bytes.

Exempel 5:

```
od -x myfile
```

Tillvalet **-x** listar med hexadecimala ord (16 bitars ord.)

HÄNVISNING

Inga

FELMEDDELANDEN

Inga

NAMN

passwd - ändra lösenord för login.

SYNTAX

passwd [-V] [-ffil] [namn]

FUNKTION

Kommandot *passwd* ändrar eller lägger in det lösenord som används vid inloggningen. Med tillvalet **-f** anges en fil som används och uppdateras istället för standardfilen **/etc/passwd**. T ex. kan lösenordet i filen **/etc/d_passwd** eller **/etc/group** ändras. **/etc/d_passwd** innehåller sekundära lösenord för uppringda förbindelser och **/etc/group** innehåller lösenord för användargrupper.

Observera: Endast användaren eller super-user kan ändra ett lösenord.

Om en vanlig användare har startat kommandot, frågar programmet först efter det gamla lösenordet, om sådant finns. Detta hindrar en obehörig att ändra ett lösenord. Därefter frågar programmet efter det nya lösenordet, två gånger. Första gången det nya lösenordet ges kollar *passwd* om det gamla lösenordet uppfyller vissa tidsvillkor inlagda av superuser. Om tidsvillkor saknas eller är uppfyllda, kollas lösenordskonstruktionen enligt följande regler. När lösenordet anges för andra gången jämförs de två lösenorden. Om de två kopiorna inte är exakt lika upprepas frågan om det nya lösenordet högst två gånger till.

Om tidsvillkoret inte är uppfyllt, avbryts kommandot. Se beskrivningen av filen **/etc/passwd** i **Systemadministation. Observera:** I filen **/etc/d_passwd** får inga tidsvillkor finnas.

Lösenord måste konstrueras enligt följande regler:

- Ett lösenord skall vara åtminstone 6 tecken långt och endast de 8 första jämföres vid inloggningen.
- Lösenordet måste ha minst två bokstavstecken och minst ett måste vara ett numeriskt tecken eller ett specialtecken. Bokstavstecken kan både vara versaler och gemener (stora och små).
- Lösenordet får inte vara samma som loginnamnet eller login-namnets tecken skiftade eller roterade. Versaler och gemener tolkas som lika vid denna jämförelsetest.
- Nya lösenord måste skilja sig från det gamla med åtminstone tre tecken. Även här tolkas versaler och gemener lika vid testen.

En super-user kan ändra alla lösenord; därför frågar *passwd* inte superuser efter det gamla lösenordet. Super-user behöver ej följa regler för lösenordens tidsvillkor eller konstruktion. En super-user kan skapa ett tomt lösenord genom att trycka på return som svar på frågan efter nytt lösenord.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *passwd*.
-ffil	Använder *fil* istället för filen **/etc/passwd**. Filen skall ha samma format på fälten som **/etc/- passwd**. Alla fält fram till och med kolon (:) efter grupp-ID måste vara korrekta i filen.

FILER

/etc/passwd	Vanliga lösenordsfilen.
/etc/d_passwd	Lösenordsfil för uppringda förbindelser.
/etc/group	Lösenordsfil för kommandot newgrp.

EXEMPEL

Exempel 1:

```
passwd
```

Systemet frågar användaren efter det gamla lösenordet. Operatören ger det gamla lösenordet och ger det nya lösenordet två gånger vid förfrågan.

Exempel 2:

```
passwd -f/etc/d_passwd kalle
```

Systemet frågar efter det gamla lösenordet för uppringda förbindelser för användaren kalle, enligt filen **/etc/d_passwd**. Det gamla lösenordet ges av operatören och det nya ges två gånger. Om användaren är superuser frågas direkt efter det nya lösenordet. Endast super-user och användaren kalle kan ändra lösenordet för kalle.

HÄNVISNING

login(1), getty(1M), newgrp(1M), su(1M)

FELMEDDELANDE

Kommandot *passwd* avslutas efter tre ej lyckade försök att ange lösenord.

ANMÄRKNING

När lösenord för uppringda förbindelser (dial-up) används, rekommenderas starkt att inte samma lösenord som vid vanlig inloggning (enligt **/etc/passwd**) används. Se även beskrivningen för *login*.

NAMN

powerfail - total avstängning av systemet

SYNTAX

Ingen (Exekveras bara av systemet och styrs av processen init och /etc/inittab).

FUNKTION

Systemprogrammet **/etc/powerfail** exekveras när datorsystemet skall stängas av totalt och strömmen skall slås av. Initieringen och den exakta funktionen av denna process beror på maskinvaran.

I vissa system kan avstängningen påbörjas automatiskt när nyckeln på framsidan vrids till OFF och/eller när man får en strömavbrottssignal från en extern kraftförsörjning.

Processen *powerfail* stänger av systemet ungefär på samma sätt som kommandot *shutdown* **-k**, men snabbare och mera direkt. Alla användare som har loggat in får 5 minuter på sig att logga ut innan de tvingas att avsluta. I vissa system slås strömmen av automatiskt när systemet har stängts av. Förutom detta finns en extra fördröjning som definieras med kommandot **/etc/mkcfig**.

TILLVAL

Inga

FILER

/etc/inittab

HÄNVISNING

init(1M), mkcfig(1M), shutdown(1M)

ANMÄRKNING

Detta kommando är systemberoende och anropas aldrig av användaren.

NAMN

pr - skriver ut filer

SYNTAX

pr [-tillval] [filnamn ..]

FUNKTION

Kommandot *pr* delar in textfiler i sidor och lägger till sidhuvud och sidfot. Överst på varje sida skrivs datum, namnet på filen eller det specificerade sidhuvudet samt sidnumret. De angivna tillvalen modifierar utskriftsformatet.

pr skriver alltid till standard output. Om standard output är en terminal förhindras alla användarmeddelanden (se kommandot *write*) medan kommandot exekveras. Om inget filnamn anges eller någon fil specificeras med ett bindestreck (-) läser kommandot *pr* standard input.

TILLVAL

De tillval som anges gäller för alla de filer som angetts om inte annat anges mellan filnamnen. **Observera:** Tillvalen kan inte nollställas mellan filnamnen men nya tillval kan läggas till. Flera tillvalsbeteckningar kan anges i en sträng, t ex *pr* **-3dh "header" fil1 fil2**.

Standardtillval är **-1 -n5 -w72 -l66**.

-V Skriver ut versionsnumret för kommandot *pr*

+*k* Börjar utskriften med sida *k*. *k* är ett siffervärde. Standard är sida 1.

-*k* Skriver ut texten i *k* antal kolumner. *k* är ett decimaltal. Den första kolumnen fylls först, sedan den andra osv. Tillvalen **-e** och **-i** antages som standard för flerkolumns utskrifter. Om inte **-s** tillvalet också ges, är kolumnerna lika breda, separerade med minst ett mellanslag, och för långa rader trunkeras. Om **-s** tillvalet anges trunkeras inte kolumnerna. Standard radlängd är 72 tecken om den inte ändras med tillvalet **-w**.

-a Skriver ut i flera kolumner horisontalt över sidan. Detta tillval måste kombineras med tillvalet **-** (t ex **-3a** for 3 kolumner). Den första raden i kolumnen fylls först, sedan den andra osv. Kolumnerna separeras och för långa rader trunkeras liksom med tillvalet **-**.

-m Skriver ut alla filer samtidigt, i var sin kolumn. Detta tillval upphäver **-*k*** och **-a** tillvalen. Det antal filer som angetts avgör antalet kolumner. Kolumnerna separeras och för långa rader trunkeras som beskrivits i tillvalet **-k**.

-d Utskrifterna görs med dubbel radframmatning mellan raderna.

-ec*n* Expanderar inkommande TAB-tecken till positionerna *n*+1, 2**n*+1, 3**n*+1 osv. Om *n* är noll eller utelämnat sätts TAB-positionerna till var åttonde position. Alla TAB-tecken expanderas till rätt antal mellanslag. Om *c* anges, används tecknet som inkommande TAB-tecken. *c* får ej vara en siffra. Om *c* utelämnas kommer TAB (011 oktalt) att användas som TAB-tecken.

-ic*n* Vid utmatning ersätts mellanslag med TAB-tecken där det är möjligt i positionerna *n*+1, 2**n*+1, 3**n*+1 osv. Om *n* är noll eller utelämnat antas var åttonde position som TAB-position. Om *c* anges (något tecken som ej är siffra) används tecknet som utgående TAB-tecken. Standard för **c** är TAB (011 oktalt).

-nc*n* Tillvalet *n* ger radnumrering med *n* siffror. Om inte *n* anges används fem siffror. Numret tar upp de *n*+1 första positionerna på varje kolumn vid normal utskrift eller på varje rad om utskriften sker med tillvalet **-m**. Om *c* anges (något tecken utom siffra) används det för att skilja numreringen från resten av raden. Anges inte *c* kommer TAB att användas (011 oktalt)

-h Tar nästa argument som ett sidhuvud, vilket anges istället för filnamnet. Om sidhuvud innehåller fler än ett ord skall det anges av apostrofer i argumentet. Tid och datum finns alltid med på sidhuvud-raden i utskriften.

-w*n* Ändrar pappersbredden till *n* tecken. Som standard används 72 tecken när man skriver ut flera kolumner, annars är standard maximal radlängd 512 tecken.

-o*n* Förskjuter utskriften av varje rad med *n* teckenpositioner (standard är 0). Antalet teckenpositioner per rad är summan av radlängden och förskjutningen.

-l*n* Ändrar sidlängden till *n* rader. Standard är 66 rader.

-p Gör en paus före utskriften av en ny sida om utskriften sker till terminal. *pr* sänder tecknet BELL till terminalen och programmet väntar på RETUR.

-r Skriver inte ut något felmeddelande om filen som skulle skrivas ut ej gick att öppna.

-t Skriver inte ut sidhuvud eller sidfot på 5 rader vardera. Fyller inte ut sista sidan av utskriften med radframmatningar till slutet av sidan.

-s(*c*) Separerar kolumnerna med tecknet *c* istället för med ett blanktecken. Standard för *c* är TAB (011 oktalt). När tillvalet **-s** används trunkeras inte raderna.

-f Separerar sidorna med tecknet för formfeed, normalt separeras sidorna med ett antal tecken för newline. Om standard output är en terminal gör *pr* en paus även före första sidan.

-b Samma som **-f** tillvalet.

FILER

/dev/tty* för att hindra meddelanden med *write*.

EXEMPEL

Exempel 1:

```
pr kap3
```

På standard output skrivs först ett 5-radigt sidhuvud ut, innehållande två blanka rader, en rad med datum, tid och sidnummer och två blanka rader. Därefter skrivs innehållet i filen ut. Om innehållet är längre än 66 rader skrivs först den 5-radiga sidfoten ut, därefter skrivs sidan 2 ut med sidhuvud.

Exempel 2:

```
pr -t kap3
```

På standard output skrivs filens innehåll ut utan något sidhuvud eller sidfot. Varje sida skrivs till standard 66 rader, utan något 'formfeed' tecken. Använd **-f** tillvalet om form-feed önskas mellan varje sida.

Exempel 3:

```
pr kap1 >kap1.pr
```

Textfilen kap1 delas in i sidor och både sidhuvud och sidfot läggs till. Resultatet placeras i filen **kap1.pr**.

HÄNVISNING

cat(1), print(1)

FELMEDDELANDEN

Felmeddelande ges om en fil inte kan läsas.

NAMN

print - formaterad utskrift till skrivare

SYNTAX

print filnamn ...

FUNKTION

print är en kommandofil, som skriver ut de angivna filerna till systemets standardskrivare genom spooler-systemet. Det kan ändras av super-user och användas som standard utskriftskommando i systemet.

Normalt sätter print sidindelningen till 112 tecken per rad och och 48 rader per sida. Form-feed-tecken (FF) används för sidframmatning. Ett sidhuvud för filen skrivs ut på varje sida, med filnamn, tid och sidnummer.

TILLVAL

Inga

FILER

Inga

HÄNVISNING

pr(1), siv(1), sh(1), lp(1), lpr(1).

ANMÄRKNING

print är en shell-script med ett av följande innehåll, beroende på vilken skrivare-spooler som är aktiv i systemet, *lp*-systemet eller *lpr*-systemet. Se kommandona *lp* och *lpr*.

```
pr -148 -b -w112 $* | lp
```

eller

```
pr -148 -b -w112 $* | lpr
```

NAMN

ps - processtatus.

SYNTAX

ps [-Vadeflv] [-u *uidlista*] [-*termlista*] [-g *grupplista*] [-p *proclista*]

FUNKTION

ps ger viss information om pågående processer. Normalt visas endast användarens egna processer. Med tillvalet **-e** fås information om alla processer medan tillvalet **-a** endast visar processer med terminaler.

Med tillvalen **-l** eller **-v** visas en lång lista med fält enligt nedan.

En kort lista visas normalt när tillvalen **-l** eller **-v** inte anges. Denna korta lista innehåller fälten **UID**, **TTY**, **PID**, **PPID** som beskrivs nedan, och även fältet **CMD**, men med upp till 39 tecken från den kommandorad som använts för att starta respektive process.

Den långa listan visad med tillvalet **-l** innehåller:

UID Processägarens användarnamn.

TTY Enhetsnamnet på den terminal som processen tillhör. (Kontrollerande terminalnod).

PID Processens ID. Detta kan användas som argument till kommandot *kill*.

PPID Förälderprocessens ID.

STATE Flaggor med anknytning till processen i formatet xxx:s, där fälten innehåller:

T - Processen spåras av annan process

L - Processen låst till operativsystemskärnan

S - Processen är swappad.

Icke aktiva fält markeras med bindestreck('-')

s-flaggan visar processens status:

R - Processen arbetar

S - Processen är i vänteläge

Z - Processen i mellanläge

P - Processen har stannat

PRI Processens prioritet; höga tal betyder låg prioritet. Jämför kommandot *nice*. Realtidsprocesser har ett R före prioriteten och anges i området 0-215.

Normala time sharing-processer anges utan bokstav före i området 0-39, vilket motsvarar systemprioritet 216-255.

MEM_SIZE	Processens storlek i minnet på formatet 'N1+N2'. N1 = Storleken av delbar kodarea i kbyte. N2 = Storleken av icke delbar kodarea i kbyte.
UTIME	Kumulativ exekveringstid för processen.
STIME	Tid sedan processen startades.
CMD	Kommandot som startade processen. På den korta listan (utan tillvalen -l och -v), innehåller detta fält upp till 39 tecken av kommandoraden inklusive kommandoargumenten.

Den långa listan som visas med tillvalet -v innehåller samma fält som ovan, men visar följande fält istället för fälten **MEM_SIZE**, **UTIME** och **STIME**:

UPAGE	Den fysiska minnes-adressen för processen, given i hexa-decimal kod.
MP_PF	Antalet MMU sid-swap sedan processen startades.
IO_PF	Antalet fysiska sid-swap till skivminnet sedan processen startades.

TILLVAL

För tillval som använder en lista som argument kan listan specificeras på två sätt:

1. En lista av identifierare separerade med komma.
2. En lista av identifierare omgiven av anföringstecken (") och separerade med komma och/eller mellanslag.

-V	Skriver ut versionsnumret för kommandot *ps*.
-a	Information om alla processer med terminaler som ej är processgruppsledare (normalt visas bara användarens egna processer).
-d	Information om alla processer utom processgrupps-ledare.
-e	Visar alla processer.
-f	Fullständig lista (genereras alltid i D-NIX)
-l	Lång lista med storleks- och tidinformation.
-v	Lång lista med swap information. Detta tillval upphäver -l tillvalet.
-t *termlista*	Visar endast information om processer kopplade till de terminaler som är angivna i *termlista*. Terminalen kan specificeras på två sätt: Terminalnamnet (t ex tty04) eller om terminalnamnet börjar med tty räcker det att ange siffrorna (t ex 04).
-p *proclista*	Visar endast information om processer vars processnummer ingår i *processlista*.

-u *uidlista* Visar endast information om process vars användar-ID eller login-namn ingår i *uidlista*.

-g *grupplista* Visar endast information om process vars processgruppsledare ingår i *grupplista*.

FILER

/dev/mem	minnet
/etc/passwd	fil med användarnamn

EXEMPEL

Exempel 1:

```
ps
```

Listar information om användarens egna processer.

Exempel 2:

```
ps -a
```

Utökar informationen med alla processer som har en terminaltillhörighet och ej är processgruppsledare.

```
ps -e
```

-e innebär alla processer, även sådana som inte har terminaltillhörighet.

Exempel 3:

```
ps -le
```

-le innebär att man listar information om alla processer i långt format. *ps* **-le** är ekvivalent med *ps* **-lae**.

Exempel 4:

```
ps -u  olle pelle
```

Visar information om alla processer som ägs av Olle och Pelle.

Exempel 5:

```
ps -t 04 console
```

Visar information om alla processer som är kopplade till tty04 eller konsolen.

HÄNVISNING

kill(1), nice(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Detta kommando är systemberoende.

Pågående processer kan förändra informationen medan *ps* arbetar. Informationen är därför bara en ungefärlig angivelse av ett systems status. I tidigare versioner av *ps* användes tillvalen **-x** och **-p** istället för nuvarande **-e** och **-v**.

PS(1)

NAMN

pwd - anger vilket bibliotek som är aktuellt.

/bin/pwd - anger vilket bibliotek som är aktuellt.

SYNTAX

pwd

FUNKTION

Kommandot *pwd* skriver ut hela sökvägen (pathname) från root till aktuellt bibliotek. Pathname skrivs på standard output.

Kommandot *pwd* är internt i shell och utförs utan att någon ny process skapas. Omdirigering av utdata kan emellertid göras.

/bin/pwd finns enbart för kompatibilitet med äldre system.

TILLVAL

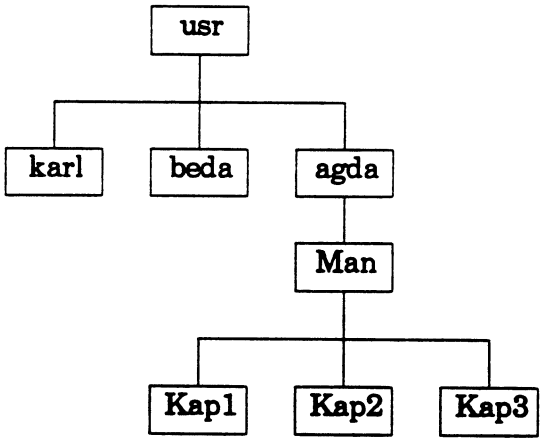
Inga

FILER

Inga

EXEMPEL

Följande trädstruktur finns i ett filsystem.



Om man befinner sig i Man och kommandot *pwd* anges blir resultatet **/usr/agda/Man**.

HÄNVISNING

cd(1)

PWD(1)

NAMN

queue - skriver en lista över de filer som finns i spoolerkön i *lpr*-systemet.

SYNTAX

queue [-V -v -t *typ*] [[-r | -p *nn*] jobnr]

FUNKTION

Kommandot *queue* skriver ut en lista över de filer som finns i spoolerkön i spooler-systemet *lpr*. Listan har följande utseende med exempel:

DEVICE	JOBID	PRI	UID	COMMENT
/dev/lp	227	40	root	Spooled ...
	228	40	bin	Spooled ...

Listan är sorterad i utskriftsordning (PRI). Det kan finnas flera enheter (device) i listan. Enheterna hamnar då i samma ordning som de är definierade i **/usr/lib/lpdpar/lpdtable.**

För att välja enstaka enheter eller vissa typer av skrivare används tillvalet **-t** (se kommandot *lpr*).

Kommandot *queue* kan användas när utskriftsbegäran (JOBID) ska tas bort från spoolerkön genom att använda tillvalet **-r**. Det går inte att ta bort andra användares utskriftsbegäran, om man inte är inloggad som super-user. Det går heller inte att ta bort utskriftsbegäran som håller på att skrivas ut, dvs den utskrift som ligger först i kön till en enhet.

Det är möjligt att sänka utskriftsprioriteten (PRI) för en utskriftsbegäran med tillvalet **-p**. Till tillvalet **-p** anger man då ett värde som motsvarar hur mycket prioriteten ska sänkas. En super-user kan använda ett negativt värde för att höja prioriteten. Utskriftsprioriteten avgör utskriftsordningen av filerna. Ett lägre PRI-värde på listan betyder högre prioritet.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *queue*.
-v	Verifierar vad *queue* utför. Används tillsammans med tillvalen **-r** och **-p**.
-t type	Anger vilken/vilka skrivare som ska visas på listan. Se **-t** tillvalet för kommandot *lpr*.
-r nn	Tar bort utskriftsbegäran från kön. Jobnummer (JOBID) anges efter **-r**.
-p nn	Minskar prioriteten för en utskriftsbegäran med nn. Jobnummer (JOBID) anges efter **-p**.

FILER

/usr/spool/lpd/*
/usr/lib/lpdpar/*

EXEMPEL

Exempel 1:

En utskriftsbegäran ska tas bort från spoolerkön. Med ett tidigare *queue* kommando har konstaterats att jobnumret är 228.

```
queue -r 228
```

Här tas utskriftsbegäran nummer 228 bort från kön.

Exempel 2:

Prioriteten på utskriftsbegäran 228 ska sänkas med 10

```
queue -p 10 228
```

HÄNVISNING

lpr(1)

FELMEDDELANDEN

Ett felmeddelande visas om det jobnummer som angetts inte finns.

ANMÄRKNING

Se *lpstat* om spooler-systemet *lp* används.

NAMN

readonly - förhindrar ändringar av shellvariabler.

SYNTAX

readonly namn ... ...

FUNKTION

Med kommandot *readonly* skyddas shell variablerna mot ändringar. Kommandofiler kan använda dessa variabler, om exporterade, men kan inte ändra dem ens temporärt.

Detta kommando är internt i shell och utförs utan att skapa en ny process.

EXEMPEL

```
HH=/usr/kurt
export HH
readonly HH
```

Shell variabeln $HH kan nu användas man kan inte ändras i shellprocedurer startade från aktuellt shell.

HÄNVISNING

sh(1), export(1)

FELMEDDELANDEN

Om man försöker ändra en *readonly* variabel, visas ett felmeddelande på skärmen:

```
'.... is  readonly'
```

READONLY(1)

NAMN

red - begränsad variant av *ed* text editor.

SYNTAX

red [-V] [-] [-p string] [file]

FUNKTION

red är en begränsad version av *ed*. Den tillåter enbart ändring av filer i det aktuella biblioteket. Den hindrar användande av !-kommandot för att starta en ny shell. Vid försök att gå förbi dessa hinder ges ett felmeddelande.

I övrigt gäller beskrivningen för kommandot *ed*.

TILLVAL

-V	Skriver ut versionnumret för kommandot *red*.
-	Undertrycker meddelanden från *ed*; t ex teckenantalet från kommandona **e**, **r** och **w** och diagnostik från kommandona **e** och **q** samt av promten ! efter ett !shell kommando.
-p ***promptsträng***	Ger användaren möjlighet att specificera en promptsträng. Standard är att ingen prompt används. Ett mellanslag krävs i kommandoraden mellan **-p** och *promptsträng*.

FILER

Inga

HÄNVISNING

ed(1), dmacs(1)

FELMEDDELANDEN

Ett felmeddelande ges vid försök att utföra förbjudna kommandon.

RED(1)

NAMN

reject - förhindrar LP utskriftsbegäran.

SYNTAX

/usr/lib/reject [-V] [-r[*orsak*]] destinationer

FUNKTION

reject hindrar *lp* från att acceptera utskriftsbegäran till angivna destinationer.

Detta kommando beskrivs mera detaljerat tillsammans med kommandot *accept*.

HÄNVISNING

accept(1M), lp(1)

NAMN

rinstall - installera nya D-NIX-versioner

SYNTAX

/etc/rinstall [-V]

FUNKTION

Kommandot *rinstall* används endast, och skall alltid användas, efter det att nya filer från leveransdisketterna med nya D-NIX versioner lästs in. De nya sysemfilerna installeras och eventuella användardefinierade systemparametrar i operativsystemsfilen kan överföras från den gamla filen innan det nya systemet aktiveras.

De flesta systemfiler och operativsystemsfilen levereras och läses in med namn som slutar med ett '+'-tecken för att undvika överskrivning av gamla filer, som kan ha ändrats av användaren i det aktuella datorsystemet.

Med kommandot *rinstall* omdöps de gamla versionerna av systemfilerna till backupfiler som slutar med tecknet '-' och de nya filerna döps om till de riktiga namnen. Kommandot är interaktivt och frågar användaren för varje fil om den ska installeras eller inte. Svaret y ska anges för att installera de nya filerna.

TILLVAL

-V Skriver ut versionsnumret för kommandot *rinstall*.

FILER

dnix (D-NIX operativsystemsfil)
/etc/brc, /etc/bcheckrc, /etc/gettydefs, /etc/inittab, /etc/motd, /etc/passwd, /etc/group, /etc/profile, /etc/rc, /etc/timezone, /etc/termcap, /usr/etc/bascap, /usr/etc/basicerr.txt, /usr/etc/sortorder.tab, /usr/etc/sorttable, /usr/etc/sorttable.f, /usr/etc/translate.txt, /usr/lib/uucp/Devices, /usr/lib/cron/cron.deny, /usr/spool/lp

Listan ovan är endast ett exempel och mera filer kan tillkomma. En komplett lista på filer finns i början av filen **/etc/rinstall**. Använd kommandot *cat /etc/rinstall* för att se den kompletta listan.

HÄNVISNINGAR

mkcfig(1M), mv(1)

ANMÄRKNINGAR

Se Revisionsinformation som levereras med alla system för versionsspecifik information.

RINSTALL(1M)

NAMN

rm, rmdir - tar bort filer, tar bort bibliotek.

SYNTAX

rm [-Vfriv] filnamn ...

rmdir [-Vps] bibliotek ...

FUNKTION

Kommandot *rm* tar bort en eller flera filer från filsystemet. Om en fil har flera länkar, förstörs inte filen förrän den sista länken tas bort. Om ett bibliotek anges istället för en fil, skrivs ett felmeddelande ut såvida inte tillvalet **-r** angetts.

För att det ska vara möjligt att ta bort en fil, måste användaren ha skrivtillstånd i det bibliotek som filen befinner sig i. Däremot behöver användaren inte ha vare sig läs- eller skrivtillstånd för filen.

När kommandot används på filer som man inte har skrivtillstånd i och om standard output är en terminal, kommer programmet att interaktivt fråga användaren om filen skall förstöras. Kommandot väntar därefter på ett svar. Om den första bokstaven i svaret är ett y kommer filen att förstöras, under förutsättning att förälderbibliotekets tillstånd tillåter detta.

Kommandot *rmdir* tar bort bibliotek. Dessa bibliotek kan bara tas bort om de är tomma.

Observera: Det är förbjudet att ta bort '..', dvs förälderbiblioteket.

Observera: Var försiktig med kommandot *rm* och använd alltid tillvalet **-i** när det är möjligt *rm* **-i** filnamn eftersom kommandot *rm* är ett kraftfullt raderingskommando.

TILLVAL

-V	Skriver ut versionnumret för kommandot *rm* eller *rmdir*.
rm	
-f	Denna flagga gäller bara filer som är skrivskyddade. Ingen fråga ställs innan filen tas bort.
-r	Inget meddelande skrivs ut om den angivna filen är ett bibliotek. Allt innehåll i biblioteket kommer att raderas liksom biblioteket själv.
-i	Frågar interaktivt för varje fil om den ska tas bort.
-ir	Frågar om varje bibliotek som påträffas ska tas bort.
-v	Visar filnamn som tas bort.
rmdir	
-p	Tar även bort alla fadersbiblioteksom blir lediga
-s	Felmeddelanden skrivs ej ut, när tillvalet **-p** används.

RM(1), RMDIR(1)

FILER

Inga

EXEMPEL

Exempel 1:

```
rm /Manual/kap8
```

Filen kap8 tas bort ur biblioteket Manual.

Exempel 2:

```
rm *
```

Tar bort alla filer i nuvarande bibliotek, utom de som börjat med punkt (.).

VARNING! Se anmärkning nedan!

Exempel 3:

```
rm -i kap*
kap1 ? [y/n]: y
kap2 ? [y/n]: n
```

Kommandot frågar interaktivt om alla filer som börjar på kap i nuvarande bibliotek skall tas bort. På standard output skrivs namnet på varje fil, som i det här fallet börjar på kap, åtföljt av **[y/n]:** Om svaret börjar med ett y kommer filen att tas bort. I alla andra fall kommer filen att behållas. I exemplet raderas filen kap1 men filen kap2 lämnas kvar.

Exempel 4:

```
rmdir mydir
```

Biblioteket **mydir** tas bort men bara om det redan är tomt. Om inte ges ett felmeddelande.

HÄNVISNING

ln(1), mv(1), scrfile(1)

FELMEDDELANDEN

Felmeddelanden ges om de givna filerna eller biblioteken inte kan tas bort.

ANMÄRKNING

VARNING! Använd inte '*' eller '.' eller '.*' som wildcardparameter till kommandot *rm* utan att först kontrollera att endast de filer som ska tas bort är inkluderade. Kontrollera **alltid** filnamnen med kommandot *l* * eller *l* . eller *l* .* innan kommandot *rm* ges med ett sådant argument.

NAMN

rmail - sända meddelande till användare (begränsat).

SYNTAX

rmail [-V -t] mottagare

FUNKTION

Med *rmail* kan man bara **sända** meddelande; *uucp*(1) använder *rmail* som säkerhetsåtgärd.

När mottagare anges tar *rmail* standard input fram till end-of-file (eller till en rad som endast innehåller en punkt "." och lägger till texten till varje mottagares meddelandefil. Meddelandet föregås av avsändarens namn och adressrader. Adressrader i meddelandet (t ex "From ...") inleds av tecknet >.

En mottagare är vanligen ett användarnamn som godtagits av *login* enligt **/etc/passwd**. Alternativt kan ett alias-namn anges, vilket av *rmail* kommer att översättas till ett verkligt mottagarnamn.

Om man sänder ett meddelande till en mottagare som inte finns i **/etc/passwd**, eller om *rmail* avbryts under inmatning av text, sparas meddelandet i filen **dead.letter** för editering och återsändning. Observera att **dead.letter** anses vara en temporär fil och återskapas varje gång som den behövs samtidigt som det gamla innehållet tas bort.

För att ange en mottagare på ett fjärranslutet system måste man sätta systemnamn och ett utropstecken före mottagarens användarnamn. Allting efter det första utropstecknet i mottagare tolkas av fjärr-systemet. Om t ex mottagare har flera utropstecken kan det betyda en sekvens av datorer genom vilka meddelandet skall sändas på väg till den slutliga destinationen. Om man t ex specificerar a!b!cde som mottagarnamn sänds meddelandet till användare b!cde i systemet a. Systemet a tolkar denna destinationsangivelse som en förfrågan att sända meddelandet till användare cde i systemet b. Det kan vara användbart t ex när avsändar-systemet kan nå systemet a men inte b men system a kan nå system b. *mail* använder inte *uucp* om det angivna fjärr-systemet är namnet på lokalsystemet (dvs. localsystem!user).

Meddelandefilen kan behandlas på två sätt för att ändra mail-funktionen. Filens åtkomsttillstånd för 'alla övriga' kan vara läs-skriv, endast läs, eller varken läs eller skriv för att medge en viss avskildhet. Om den ändras till annat än standard kommer filen inte att tas bort när den är tom för att behålla de önskade åtkomsttillstånden. Första raden i filen kan också innehålla texten:

```
Forward to nymottagare
```

som då gör att alla meddelanden som skickas till ägaren av filen sänds vidare till 'nymottagare'. Det är mycket användbart när man ska skicka alla meddelanden avsedda för samma person till en viss maskin i en organisation med flera anslutna datorer. För att vidarebefordran skall fungera bör mail-filen ha "mail" som grupp-ID, och åtkomsttillstånden för gruppen bör vara läs/skriv.

När en användare loggar in ges en notering om det finns något meddelande. En notering ges också om meddelanden kommer in medan man använder kommandot *mail*.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *rmail*.
-t	I början av meddelandet läggs till en rad med namn på alla mottagare som meddelandet sänds till.

FILER

/etc/passwd	att identifiera avsändare och lokalisera mottagarpersoner.
$MAIL	variabel som innehåller pathname till mottagarens mail-fil.
/tmp/ma*	temporär fil.
/usr/mail/*.lock	temporär åtkomstspärr för mailbiblioteket.
dead.letter	text som ej kan sändas

HÄNVISNING

login(1), mail(1), write(1)

ANMÄRKNING

Olika förhållanden kan ibland innebära att spärrfilen inte kan tas bort.

NAMN

rmuser - tar bort en användare

SYNTAX

/etc/rmuser [-V] [-rn] [användarnamn]

FUNKTION

rmuser tar bort en användare genom att uppdatera filerna **/etc/passwd** och **/etc/group** samt genom att ta bort användarens hembibliotek. Om användarnamnet ej anges frågar kommandot efter det. Användarens hembibliotek tas bort endast om det är tomt, detta kan dock kringgås om man använder tillvalet **-r**. När man har tagit bort användaren undersöker kommandot filen **/etc/group** och tillåter operatören att interaktivt ta bort (eller behålla) grupper, som ej har några medlemmar.

Observera: Endast super-user får använda detta kommando.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *rmuser*.
-r	Tar bort användarens hem-bibliotek även om det ej är tomt. Allt innehåll i biblioteket kommer att försvinna, även underbibliotek.
-n	Tar ej bort användarens hembibliotek.

FILER

/etc/passwd
/etc/group

EXEMPEL

Exempel 1:

```
/etc/rmuser
```

Kommandot frågar efter användarnamn, varefter användaren kommer att tas bort. Hembiblioteket tas bort endast om det är tomt.

Exempel 2:

```
/etc/rmuser -n olle
```

Användaren olle tas bort. Dock kommer olles hembibliotek att lämnas kvar.

HÄNVISNING

mkuser(1M)

FELMEDDELANDEN

Inga

RMUSER(1M)

NAMN

rsh - anropar en begränsad shell kommandotolk.

SYNTAX

rsh [-ceiknrstuvxafhV] [argument]

FUNKTION

Programmet *sh* är ett standard programspråk som utför kommandon lästa från en terminal eller fil. Normalt startas det automatiskt av systemet för varje användare som loggar in.

rsh kommandot är detsamma som att använda ***sh*** med tillvalet **-r** (restricted shell) vilket begränsar användaren på olika sätt.

För detaljer se beskrivningen av kommandot *sh*.

RSH(1)

NAMN

scrfile - tar bort filer och rensar arean på skivminnet

SYNTAX

scrfile [-Vf] filnamn ...

FUNKTION

Kommandot *scrfile* tar bort en eller flera filer från filsystemet och rensar skivminnet där filerna fanns. Den platsen fylls med ASCII värdet 0 och systembufferten tvingas ner till skivminnet (flush) innan filen tas bort. Jämför kommandot *rm* som endast tar bort filen från biblioteksstrukturen.

En fil med flera länkar kan inte tas bort utan tillvalet **-f** och ett felmeddelande visas som anger antalet länkar till filen.

Med tillvalet **-f** fylls alltid motsvarande area med ASCII 0 och länken till filen tas bort. Om mer än en länk finns ändras inte de andra länkarna men filens innehåll raderas.

För att kunna ta bort en fil måste användaren ha skrivtillstånd både till filen och det bibliotek där filen finns.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *scrfile*.
-f	Tvingar rensning även av filer med flera länkar.

FILER

Inga

EXEMPEL

Exempel 1:

```
scrfile /Manual/kap8
```

Filen kap8 tas bort ur biblioteket Manual.

Exempel 2:

```
scrfile *
```

Tar bort alla filer i aktuellt bibliotek utom de som börjar med en punkt (.).

VARNING! Se noteringen nedan.

HÄNVISNINGAR

rm(1), rmdir(1)

FELMEDDELANDEN

Felmeddelanden ges om angivna filer och bibliotek inte kan tas bort.

ANMÄRKNING

Med kommandot *scrfile* rensas filen fysiskt på skivminnet. Med kommandot *rm* kan inte garanteras att detta sker även om man skriver värdet

0 till filen innan *rm*-kommandot, eftersom innehållet i filbufferten inte skrivs ner fysiskt till skivminnet om det inte tvingas ner med ett särskilt systemanrop (flush).

VARNING! Använd inte '*' eller '.' eller '.*' som wildcard-parameter till kommandot *scrfile* utan att vara säker på att endast de filer som avses tas bort. Kolla därför alltid filnamnen med kommandot *l* * eller *l* . eller *l* .* innan kommandot *scrfile* anropas.

NAMN

sed - stream editor

SYNTAX

sed [-n] [-e script] [-f sfil] [filer]

FUNKTION

sed kopierar och editerar namngivna filer (vanligtvis från standard input) till standard output enligt kommandon i ett script (en kommandofil). Med tillvalet **-f** kommer *sed* dessutom att hämta kommandon från filen **sfil**. Om både **-e** och **-f** används kommer kommandon från båda filerna att användas. Om enbart tillvalet **-e** skall användas och inte **-f** behöver inte **-e** skrivas ut. Tillvalet **-n** (no print) gör att *sed* inte skriver ut behandlade rader såvida inte instruktionen **p** används. Ett script består av editeringskommandon, ett per rad, på följande form:

(adress (, adress)) funktion (argument)

Normalt kopierar *sed* in en rad i taget (såvida det inte finns någonting kvar efter ett D-kommando) till ett område för modifikation, ett s.k. "pattern space", och utför sekventiellt de kommandon vars adress pekar ut den inlästa raden. När alla kommandon i scriptfilen har gåtts igenom kopierar *sed* den resulterande raden till standard output (utom under **-n**).

Vissa kommandon sparar undan hela eller delar av den ursprungliga inraden till ett s.k. "hold space" för att den skall kunna användas flera gånger.

En adress är antingen ett decimalt tal vilket anger radnumret i infilen, ett '$'-tecken som anger sista raden i infilen eller en s.k. /regular expression/, RE (enligt samma regler som i *ed* modifierad enligt:

1. I en s.k. context adress är konstruktionen \?regular expression?, där ? kan vara vilket tecken som helst, identiskt med /regular expression/ (RE). Observera att i context adressen \xabc\xdefx står det andra x för sig självt så att RE är lika med abcxdef.
2. Escape-sekvensen \n matchar nyrad-tecknet i en inläst rad.
3. En punkt (.) matchar varje tecken utom nyrad-tecknet.
4. En kommandorad utan adress exekveras på alla inlästa rader.
5. En kommandorad med en adress exekveras på alla inlästa rader som matchar adressen.
6. En kommandorad med två adresser exekveras på de rader som innefattas i området från den första rad som matchar den första adressen till och med den rad som matchar den andra adressen. (Om den andra adressen är mindre eller lika med den första väljs endast en rad ut). Därefter upprepas detta genom att på nytt söka efter den första adressen.

Genom att använda funktionen ! (nedan) kan man få kommandon att exekveras endast på de icke utvalda raderna.

I följande lista över funktioner indikeras maximalt antal tillåtna adresser för varje funktion innanför parenteser.

Textargumentet består av en eller flera rader, där alla utom den sista slutar med \ för att dölja nyrad-tecknet. Bakåtlutande snedstreck (\) i text behandlas på samma sätt som bakåtlutande snedstreck i ersättningssträngen i ett s-kommando och kan användas för att skydda inledande mellanslag och tab-tecken från den avskalning som alltid utförs på raderna i ett script. Argumenten rfil och wfil måste avsluta kommandoraden och föregås av ett mellanslag. Varje wfil skapas innan exekveringen börjar. Det kan som mest finnas 10 wfil argument.

(1)a\ *text*	Append. Skickar *text* till utenheten innan nästa rad läses in.
(2)b *label*	Hoppar till kommandot : där *label* finns. Om label saknas hoppar du till slutet på kommando-filen.
(2)c\ *text*	Change. Raderar "pattern space". Om två adresser anges kommer hela adressområdet att ersättas av *text*. *text* skickas till utenheten. Startar nästa inläsningscykel.
(2)d	Raderar "pattern space". Startar nästa inläsningscykel.
(2)D	Raderar första delen av "pattern space" fram till det första nyrad-tecknet. Startar nästa inläsningscykel.
(2)g	Ersätt innehållet i "pattern space" med innehållet i "hold space".
(2)G	Lägger till innehållet i "hold space" till "pattern space".
(2)h	Ersätt innehållet i "hold space" med innehållet i "pattern space".
(2)H	Lägger till innehållet i "pattern space" till "hold space".
(1)i\ *text*	Insert. Skickar *text* till standard output.
(2)l	Skickar innehållet i "pattern space" till standard output. Icke skrivbara tecken skrivs ut med ASCII-kod och långa rader delas upp i mindre.
(2)n	Kopierar "pattern space" till standard output. Ersätter "pattern space" med nästa indatarad.
(2)N	Lägger till nästa inlästa rad till "pattern space" (avskiljt med ett nyrad-tecken). Aktuellt radnummer ändras.
(2)p	Print. Kopierar "pattern space" till standard output.
(2)P	Kopierar första delen (fram till det första nyradtecknet) i "pattern space" till standard output.
(1)q	Quit. Hoppar till slutet på kommandofilen (scriptet). Startar inte en ny cykel.
(2)r *rfil*	Läser innehållet i *rfil* och skickar det till utenheten innan nästa rad läses in.

(2)s/RE/ersättningssträng/

flagga Ersätter RE med ersättningssträngen på utvalda rader. Valfritt tecken kan användas istället för '/'-tecknet. För en utförligare beskrivning se *ed*(1). Flaggan kan vara något av följande:

n n är ett nummer mellan 1 och 512.

Ersätter endast den n:te förekomsten av RE.

g Global.Ersätter alla icke överlappande förekomster av RE istället för endast den första förekomsten.

p Skriver ut "pattern space" om en ersättning utfördes.

w *wfil* Write. Lägger till "pattern space" till filen *wfil* om en ersättning utfördes.

(2)t *label* Test. Hoppa till kommandot : där *label* finns om någon ersättning har gjorts sedan den senaste inläsningen av en rad eller exekveringen av instruktionen t. Om *label* saknas, hoppa till slutet på kommandofilen (scriptet).

(2)w *wfil* Write. Skickar "pattern space" till filen *wfil*.

(2)x Växlar innehållen i "pattern space" och "hold space".

(2)y/*string1*/*string2*/

Transform. Byter ut alla tecken i *string1* mot motsvarande tecken i *string2*. *string1* och *string2* måste vara av samma längd.

(2)! *function* Negation. Exekvera *function* (eller gruppen av funktioner om *function* är ä) endast på de rader vilka inte valts ut av adressen (adresserna).

(0): *label* Detta kommando gör ingenting. (Det innehåller dock en *label* som kommandona **b** och **t** kan hoppa till).

(1)= Skickar aktuellt radnummer till standard output som en rad.

(2){ Exekverar följande kommandon fram till tecknet } på utvalda rader.

(0) Ett tomt kommando ignoreras.

(0)# Om ett #-tecken dyker upp som det första tecknet på den första raden i ett script betraktas hela raden som en kommentar, med ett undantag. Om tecknet efter #-tecknet är ett 'n' betyder detta att standard output inte kommer att användas. Resterande del av raden efter #n ignoreras. Ett script måste innehålla minst en icke-kommentars-rad.

HÄNVISNINGAR

awk(1), ed(1), grep(1)

SED(1)

NAMN

set - listar eller ändrar shellvariabler

SYNTAX

set [[+ | -]ekntuvxafh-] [arg ... ...]

FUNKTION

Kommandot *set* utan argument eller tillvalsflaggor listar alla definierade shellvariabler.

Tillvalsflaggorna ges som tillval i en sträng och de ändrar shellomgivningen (environment) enligt listan nedan. Dessa flaggor kan också ges som tillval vid start av en ny shell. Föregången av ett bindestreck '-' som i listan nedan aktiveras funktionen. Föregången av ett plustecken '+', stängs funktionen av.

Argumenten arg1 arg2 ..... ges till shell positionsparametrar $1, $2, ... , och ersätter de parametrar som existerade vid starten av nuvarande shell.

Kommandot *set* är internt i shell och utförs utan att en ny process skapas. Ingen omdirigering av in- och utdata är tillåten.

TILLVAL

-e	Nuvarande shell avslutas omedelbart om ett kommando returnerar utgångsstatus skilt från noll. Tillvalet **-e** har ingen effekt om shell är interaktivt.
-k	Alla shellvariabler som definieras på en kommandorad exporteras till kommandot, inte endast de som kommer före kommandot på kommandoraden.
-n	Läser kommandon men utför dem inte. Användbar tillsammans med flaggan **-x**. Inte lämplig i interaktivt mod.
-t	Avslutar aktuellt shell när ett kommando lästs och utförts. Inte lämplig i interaktivt mod.
-u	Variabler som inte är definierade behandlas som fel när substitution sker. Normalt behandlas en odefinierad variabel som en tom sträng.
-v	Skriver ut shell input-rader när de läses.
-x	Skriver ut kommandon och argument när de utförs.
-a	Markerar för export alla variabler som ändras eller skapas.
-f	Omöjliggör generering av filnamn enligt 'wildcards'.
-h	Avkodar och minns funktionskommandon då funktionen definieras. Normalt avkodas de först när funktionen utförs.

-- Detta är inte ett tillval. Det avkodas som ett argument och avgränsar tillvalen som gör det möjligt för argument som följer att börja med bindestreck (-) utan att avkodas som tillval.

EXEMPEL

Exempel 1:

```
set AAA BBB CCC
```

Detta kommando omdefinierar positionsparametrarna $1 to AAA, $2 till BBB och $3 to CCC.

Exempel 2:

```
set -x
```

Detta shell kommer nu att eka alla kommandorader till standard output. Funktionen kan tas bort med kommandot:

```
set +x
```

HÄNVISNING

sh(1), unset(1)

ANMÄRKNING

Förutom ovanstående tillval kan tillvalen **-c**, **-s**, **-i** och **-r** ges då kommandot *sh* startas. Även dessa listas med *set* men kan inte sättas eller ändras med *set*.

NAMN

setmnt - uppdatera filerna **/etc/mnttab, /etc/mtab.**

SYNTAX

/etc/setmnt [-V]

FUNKTION

Kommandot *setmnt* läser information från standard input för att skapa och uppdatera filerna **/etc/mnttab** och **/etc/mtab. /etc/mtab** uppdateras endast om den redan finns i systemet. *setmnt* används huvudsakligen när operativsystemet startas och tillsammans med kommandot **/etc/devnm** för att skapa tabellfilerna med information om enheten root samt om monterade filhanterare. Filerna används också av kommandona *mount* och *umount.*

TILLVAL

-V	Skriver ut versionsnumret för *setmnt.*

FILER

/etc/mnttab	Monterade D-NIX filsystem
/etc/mtab	Monterade hanterare

HÄNVISNING

devnm(1M), mount(1M), umount(1M)

/dev/brc

SETMNT(1M)

NAMN

setspeed - ändrar temporärt terminalparametrar för seriella portar.

SYNTAX

/etc/setspeed [-V] [-*in*] [-*on*] [-*sn*] enhet ...

/etc/setspeed [-V] -h parametrar < enhet

FUNKTION

Kommandot *setspeed* sätter hastigheten (baudrate) och övriga kommunikationsparametrar för seriella enheter utan att man behöver vara inloggad. I första formatet ovan sätts enbart baudrate medan andra formatet (**-h**) används om även andra parametrar skall ändras.

Kommandot läggs normalt i filen **/etc/rc** eller **/etc/inittab** för automatisk aktivering vid systemstart.

setspeed är särskilt lämpligt för skrivaranslutningar. *setspeed* ligger kvar som en "demon" och skillnaden mot att använda kommandot *stty* är att de ursprungliga parametrarna återställs då *setspeed* tas bort.

Om *setspeed* används mot en terminal med inloggningsmöjlighet, upphör parametrarna dessutom att gälla vid utloggning eftersom inloggningsprogrammet alltid återställer parametrarna vid en normal inloggningsprocedur.

Man kan blanda hastigheter och enheter i uttrycket, som t ex:

```
/etc/setspeed -o2400 /dev/lp -s1200 /dev/letter &
```

Detta sätter hastigheten ut till **/dev/lp** till 2400 baud. Hastigheten till och från **/dev/letter** sätts till 1200 baud. För skrivare som använder XON/XOFF-signalering skall normalt in- och uthastigheten vara lika.

Om felaktiga parametrar anges avbryts programmet och parametrarna återställs.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *setspeed*.
-*in*	Sätter in-hastigheten 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.
-*on*	Sätter ut-hastigheten till *n*, där *n* väljs på samma sätt som beskrivs för **-*in***.
-*sn*	Sätter in- och uthastigheterna till *n*, där *n* väljs på samma sätt som beskrivs för **-*in***.
-h	Alla parametrar inklusive enhet ska anges exakt som i kommandot *stty*. Observera att tecknet < behövs och ett mellanslag mellan **-h** och parametrarna. Exempel: **/etc/setspeed -h 2400 </dev/tty05**. Se *stty* för de parametrar som ska användas.

FILER

/dev/* Fysiska terminalportar

HÄNVISNING

stty

FELMEDDELANDEN

Inga

ANMÄRKNING

Olika hastigheter för in- och utdata (split speed) kan enbart användas om enheten klarar av detta samt om programvarorna kan hantera split speed.

NAMN

sh, rsh - anropar shell kommando-tolk.

SYNTAX

sh [-ceiknrstuvxafhV] [argument]

rsh [-ceiknrstuvxafhV] [argument]

FUNKTION

Programmet *sh* är ett standard program-språk som utför kommandon lästa från en terminal eller en fil. Vanligen startas *sh* automatiskt av systemet för varje användare efter loginproceduren.

Kommandot *rsh* (restriktiv eller begränsad shell) är samma som att använda kommandot *sh* med tillvalet **-r** och begränsar användaren på olika sätt. Användaren av en restriktiv shell får inte byta bibliotek, ändra PATH-variabeln eller specificera något filnamn eller kommandonamn som innehåller tecknet /. Omdirigering av utdata med > eller >> kan inte göras.

Vad tillval och argument betyder förklaras nedan under avsnittet START AV SHELL.

SHELLKOMMANDON

Ett enkelt kommando är en rad icke blanka ord separerade av blanktecken. Det första ordet anger kommandots namn. Förutom vad som sägs nedan, skickas övriga ord som argument till det startade kommandot. Shell returnerar kommandots utgångsstatus. Om kommandot inte har avslutats på normalt sätt blir utgångsstatus 200+statusvärdet (oktalt), dvs om kommandot dumpats till filen core.

En pipeline är en sekvens av ett eller flera kommandon åtskilda av tecknet |. Standard output från varje kommando, utom det sista, ansluts via en pipe till standard input för nästa kommando. Varje kommando körs som en separat process; shell väntar på att det sista kommandot ska avslutas.

En lista är en sekvens av en eller flera pipelines åtskilda av ;, &, && eller ||, och eventuellt avslutade med ; eller &. Symbolerna ; och & har lika rang, vilken är lägre än rangen för && och ||. && och || har också lika rang. Ett semikolon(;) ger sekvensiell exekvering av pipelines; tecknet & ger asynkron exekvering av föregående pipeline i bakgrunden (dvs shell väntar inte på att denna skall avslutas innan nästa startas).

Symbolen && anger att kommandolistan som följer utförs endast om föregående pipeline ger värdet 0 som utgångsstatus. Symbolen || anger att den utförs om utgångsstatus är skilt från 0. Det semikolon (;) som separerar kommandona kan ersättas med 'newline'.

Ett shell-kommando är antingen ett enkelt kommando eller något av de följande speciella kommandona. I beskrivningen betyder beteckningen 'lista' en kommandolista och utgångsstatus från kommandolistan är samma som utgångsstatus från det sista kommandot i listan om inget annat anges. Istället för ';' kan ett newline-tecken användas, dock inte när dubbelt ';;' används.

for namn (in ord1 ord2 ...) ; do lista; done

Varje gång kommandona i lista utförs, sätts variabeln namn till nästa ord i ordlistan. Om ordlistan utelämnas, utförs kommandolistan en gång för varje positionsparameter på kommandoraden, dvs namn är $1, $2, ... (Se PARAMETERSUBSTITUTION nedan). Bearbetningen avslutas när inga ord återstår i ordlistan eller om kommandot *break* påträffas.

Kommandofilen i exemplet nedan kommer att skriva ut alla filer, som givits som argument, med filnamnet före varje fil:

```
for i
do echo 'Filnamnet är:' $i ; cat $i
done
```

case ord in (mönster (| mönster) ...) lista ;;).... esac

Kommandot *case* bearbetar listan som följer det första mönstret, där mönster matchar värdet av $ord. Flera mönster kan anges före varje lista. Observera ')' mellan mönstren och den motsvarande kommandolistan. Mönstrets form är densamma som används för att skapa filnamn. Tecknen *, ?, [,] och ! har speciell betydelse. (Se SKAPANDE AV FILNAMN nedan).

Kommandofilen i exemplet nedan visar hjälpinformation om första argumentet $1 är -V.

```
case $1 in
   -V) echo 'Hjälp-text exempel' ;;
   *) echo 'Utför kommandot!' ;;
esac
```

if lista ; then lista ; (elif lista ; then lista ;) ... (else lista ;) fi

Listan efter *if* utförs och om den ger utgångsstatus noll utförs enbart listan efter det första *then*. Annars utförs listan efter *elif*, och om nya utgångsstatusen är noll, utförs även listan efter det andra *then*. Däremot, om utgångsstatus från den första listan efter *if* är icke-noll utförs listan efter det sista *else*. Om en *else*-lista finns angiven och ingen then lista utförs ger hela if-kommandot utgångsstatus noll. Kommandot *fi* avslutar kommandot *if*.

Kommandofilen i följande exempel avslutas med ett meddelande om antalet argument på kommandoraden ($#) är mindre än 3.

```
if test $# -lt 3
   then echo 'För få argument.' ; exit
fi
```

while lista ; do lista ; done

Kommandon i listan *while* utförs. Om utgångsstatus är icke-noll utförs *do*-listan och sedan utförs *while*-listan igen. Detta fortsätter tills kommandolistan efter *while* ger utgångsstatus noll eller tills kommandot *break* påträffas i listan. Utgångsstatus från *while* kommandot är alltid noll.

until lista ; do lista ; done

Detta kommando har samma funktion som *while*, men kommandolistorna utförs upprepade gånger tills utgångsstatus är icke-noll.

(lista)

Kommandolistan inom parenteser bearbetas i en ny shell, som kan använda samma omdirigering av in- och utdata.

```
( echo 'titel' ; cat fil ) > listfil
```

{ lista ; }

Detta är samma som att utföra kommandona i listan. Kommandona bearbetas i nuvarande shell. Observera den slutliga ';', som krävs efter listan när {} parentesen används. Tecknet ';' kan också ersättas med ett newlinetecken.

namn () { lista ; }

Detta kommando definierar en namngiven funktion. För att utföra kommandona i lista anges endast funktionsnamnet och kommandona utförs inom nuvarande shell utan att en ny process skapas. Den avslutande ';' i listan kan ersättas med ett newlinetecken. Medan bearbetning pågår sätts positionsparametrarna $1, $2 ... till funktionens argument.

Följande ord avkodas endast som första ord i ett kommando och när de inte placeras inom apostrofer.

```
if then else elif fi case esac for while
until do done { } (
```

KOMMENTARER

Ett ord som börjar med "#", och alla följande ord fram till radslut, ignoreras. Jämför också specialkommandot : (kolon).

KOMMANDOSUBSTITUTION

Standard output från ett kommando inom ett tecknen '...' kan användas som del av ett ord eller som helt ord; newline på slutet tas automatiskt bort.

PARAMETERSUBSTITUTION

Tecknet $ används på en kommandorad för att införa parametrar som skall ersättas av sitt värde (substitueras) när kommandot utförs. Positionsparametrar kan tilldelas värden med *set*. Andra shellvariabler kan sättas genom att skriva shellkommandot:

```
namn=värde ( namn=värde ) . . .
```

Eventuella metatecken i värdesträngen avkodas inte till filnamn.

Normalt ersätts (substitueras) shellparametrar på en kommandorad med sina värden (en sträng), men detta kan ändras om något av följande uttryck används. I beskrivningen nedan kan ord antingen vara en sträng, en annan shellparameter eller ett substituerat kommando.

ord nedan utvärderas inte om det inte används som substituerad sträng. I följande exempel utförs kommandot *pwd* enbart om parametern DD inte redan är definierad. Formatet beskrivs på nästa sida.

```
echo ${DD-'pwd'}
```

Ett parameternamn är en sträng av bokstäver, siffror och understrykningstecken, eller en av de speciella shell-parametrarna som listas nedan. Ett

användardefinierad parameternamn måste börja med en bokstav eller ett understrykningstecken.

$parameter

${parameter} Parameterns värde, om den har något, substitueras. Tecknen { } behövs bara när parametern följs av en bokstav, siffra eller ett understrykningstecken som inte skall tolkas som en del av dess namn. Om parametern är en siffra så är den en positionsparameter. Alla positionsparametrar substitueras om parametern är * eller @. Substitueringen börjar med $1. Parametern $0 sätts från argument noll (kommandonamnet) när shell anropas.

${parameter:-ord}

Om parametern har definierats och inte är noll, substitueras dess värde; annars substitueras ord.

${parameter-ord}

Om parametern har definierats, substitueras dess värde; annars substitueras ord.

${parameter:=ord}

Om parametern inte har definierats eller är noll, sätts dess värde till ord; parameterns värde substitueras därefter. Positionsparametrar kan inte tilldelas värden på detta sätt.

${parameter=ord}

Om parametern inte har definierats, sätts värdet till ord. Parameterns värde substitueras därefter. Positionsparametrar kan inte tilldelas värde på detta sätt.

${parameter:?ord}

Om parametern har definierats eller inte är noll, substitueras dess värde; annars skrivs ord till standard output och shell avbryts. Om ord utelämnas visas meddelandet "parameter null or not set".

${parameter?ord}

Om parametern har definierats, substitueras dess värde; annars skrivs ord till standard output och shell avbryts. Om ord utelämnas visas meddelandet "parameter null or not set".

${parameter:+ord}

Om parametern har definierats och inte är noll, substitueras ord; annars substitueras en tom sträng.

${parameter+ord}

Om parametern har definierats, substitueras ord; annars substitueras en tom sträng.

Följande parametrar definieras automatiskt av shell:

#	Antal positionsparametrar, såsom ett decimalt tal.
0	Positionsparametern 0, dvs kommandonamnet självt.
1	Positionsparametern 1, dvs den första kommandoparametern.
*	En sträng med alla kommandoparametrar $1 $2 $3 ....
@	$@ är samma som $*, men behandlas olika när den är mellan citationstecken "$@". Se CITERING nedan.
-	Tillval som givits när sh startades, eller av kommandot set.
?	Det decimala värdet för utgångsstatus från det senaste synkront utförda kommandot.
$	Processnumret för nuvarande shell.
!	Det senast startade bakgrundskommandots processnummer.

Följande parametrar används av shell:

HOME	Standardargument (hembiblioteket) till kommandot *cd*. Sätts automatiskt vid inloggningsproceduren.
PATH	Sökväg för kommandon. Det är en med ':' separerad lista över bibliotek. (Se BEARBETNING nedan). Parametern PATH kan inte ändras från en restriktiv shell.
CDPATH	Sökväg för kommandot *cd*. Den konstrueras på samma sätt som parametern PATH. Om CDPATH inte är definierad, sökes enbart i det aktuella biblioteket.
MAIL	Sätts denna parameter till namnet på mail-filen kommer shell att kolla denna fil och informera användaren när post anländer varje gång som shell startas i interaktiv mod, t ex vid login eller när kommandot sh ges utan kommandoparameter. MAIL definieras automatiskt vid login till /usr/mail/user-name.
MAILCHECK	Antalet sekunder (standard 600 = 10 minuter) mellan automatisk test om post har kommit. Om MAILCHECK satts till 0, testar shell eventuella meddelanden varje gång som shell-prompten skrivs ut.
MAILPATH	Om denna parameter definierats skall den innehålla en lista över mail-filer, i vilka shell söker efter meddelanden. Det är en ':'- separerad lista av filer. Ett filnamn kan följas av tecknet % och ett meddelande, vilket skrivs ut när filen ändras, t ex när post har anlänt. Utan angiven meddelandesträng ges standardmeddelandet: `you have mail`
PS1	Primär promptsträng, standard är "$".

PS2	Sekundär promptsträng; standard är ">".
IFS	Interna fältseparatorer, normalt mellanslag, tab och newline.
SHACCT	Om denna parameter är definierad anger den en fil, skrivbar för användaren, till vilken shell lägger till ett administrativt meddelande för varje shellprocedur som utförs.
SHELL	Om denna parameter är definierad och innehåller bokstaven r när kommandot *sh* anropas, blir shell en restriktiv shell som om den hade startats med tillvalet -r.

Shell ger standardvärden till PATH, PS1, PS2 och IFS. HOME och MAIL sätts inte av shell, däremot sätts de av *login*.

TOLKNING AV BLANKTECKEN

Efter parameter- och kommandosubstitution genomsöks resultaten efter interna fältseparatorer (dvs de som återfinns i IFS), och delning till entydiga argument görs där sådana separatorer finns. Uttryckligen angivna tomma strängar ("" eller '') behålls. Implicita nollargument (dvs sådana som är resultatet av parametrar utan värde) tas bort om inte parametern står mellan "..." eller '...'.

SKAPANDE AV FILNAMN

Efter substitution söks metatecknen *, ? and [i alla kommandoargument. Finns något av dessa tecken betraktas ordet som ett filmönster. Ordet ersätts då med en lista med alfabetiskt sorterade filnamn som matchar mönstret. Om inget matchande filnamn hittas, lämnas ordet kvar som argument. Tecknet '.' i början av ett filnamn, eller omedelbart efter ett /, liksom tecknet / själv, måste anges uttryckligen.

Metatecknen och deras mönster ser ut som följer:

*	Matchar alla strängar, även tomma strängar.
?	Matchar ett tecken, vilket som helst.
[.....] eller [!....]	Matchar ett tecken, vilket som helst av de ingående tecknen. Ett par tecken åtskilda av - (minustecken) matchar alla tecken alfabetiskt mellan de angivna tecknen (inklusive tecknen själva). Om det första tecknet efter det första [är ett utropstecken '!', matchas istället alla tecken som inte är angivna.

CITERING

Följande tecken har speciell betydelse för shell och avslutar ett ord om tecknen inte är citerade med någon av nedanstående metoder.

```
; & ( ) | ^ < > newline space tab
```

Ett tecken kan citeras (dvs stå för sig självt) genom att man låter ett \ föregå tecknet. Paret \newline ignoreras. Alla tecken som placeras mellan apostrofer (single quote på engelska) '...', utom en apostrof citeras. Även inom citationstecknen ("...") (double quote) sker parameter- och kommando-

substitution. Mellan citationstecken ("...") krävs ett \ för citering av tecknen \, ', " och $.

"$*" är detsamma som "$1 $2 . . .", medan "$@" är detsamma som "$1" "$2"...

PROMPTER

När shell används interaktivt visas värdet av PS1 som prompt innan kommandon kan ges från terminalen. Om en rad avslutas med new-line trots att med indata krävs för att fullfölja ett kommando, visas den sekundära prompten (dvs värdet av PS2).

OMDIRIGERING AV IN- OCH UTDATA

Innan ett kommando utförs, kan dess indata och utdata omdirigeras med särskilda uttryck som tolkas av shell. Nedan listade uttryck kan förekomma var som helst i ett enkelt kommando eller kan föregå eller följa efter ett kommando. Uttrycken tolkas av shell och skickas inte vidare som argument till det anropade kommandot; substitution sker innan ord eller siffra används:

Omdirigering av utdata tillåts ej i en restriktiv shell. Se tillvalet **-r** till *sh*.

<ord	Använder filen ord som standard input (filbeskrivare 0).
>ord	Använder filen ord som standard output (filbeskrivare 1). Om filen inte existerar så skapas den; annars trunkeras den till längden noll.
>>ord	Använder filen ord som standard output (filbeskrivare 1). Om filen existerar läggs utdata till i slutet av filen (genom att söka upp end-of-file); annars skapas filen.
<<ord <<-ord	Shell indata läses fram till en rad består endast av ord, eller till en end-of-file markering. Det resulterande dokumentet blir standard input. Om något tecken i ordet ord är citerat, görs ingen tolkning av tecknen i dokumentet; annars utförs parameter- och kommandosubstitution, \newline ignoreras och \ måste användas för att citera tecknen \, $, ' och det första tecknet i ord. Om <<- används, tas alla inledande tab-tecken bort från ord och från alla dokumentrader.
<&siffra	Filbeskrivare siffra kopieras och standard input (normalt filbeskrivare 0) läses från filen associerad med denna nya filbeskrivare. Motsvarande gäller standard output om man använder tecknet >.
<&-	Stänger standard input. Motsvarande gäller för standard output om man använder tecknet >.

Om något av de ovanstående uttrycken föregås av en siffra, får den skapade filbeskrivaren det värde som anges av siffran (istället för standardvärdena 0 and 1). Exempel:

```
..... 2>&1
```

Filbeskrivare 2 skapas som en kopia av filbeskrivare 1, dvs standard error (2) skickas till samma fil som standard output.

Shell utvärderar all omdirigering som görs på kommandoraden från vänster till höger. I exemplet nedan associeras standard output (filbeskrivare 1) först med en fil, sedan associeras standard error output (filbeskrivare 2) med samma fil. Däremot, om 2>&1 hade gjorts först, skulle den sända standard error till terminalen eftersom filbeskrivare 1 som standard är riktad till terminalen.

```
..... >file 2>&1
```

Om ett kommando följs av & för att utföra kommandot i bakgrunden blir normalt den tomma filen **/dev/null** standard input för kommandot. I övrigt innehåller omgivningen för kommandot filbeskrivningarna från nuvarande shell, eventuellt modifierade av omdirigering av in- eller utdata.

OMGIVNING

En omgivning (environment) är en lista med namnvärdepar som har formuleringen namn=värde. Denna lista skickas till ett startat program på samma sätt som en vanlig argumentlista. Shell påverkar/påverkas av omgivningen på flera olika sätt. När shell startas söker den igenom omgivningen och skapar en parameter för varje namn som hittas och ger den motsvarande värde. Kommandon som därefter utförs "ärver" samma omgivning. Om användaren ändrar värdena på dessa parametrar, eller skapar nya, påverkar detta inte omgivningen om inte kommandot *export* används för att koppla shells parametrar till omgivningen. Jämför även kommandot *set* **-a**. Kommandot *unset* används för att ta bort en parametrar från omgivningen.

Omgivningen för det kommando som utförs består följaktligen av de icke-modifierade par av name=värde som från början ärvdes av shell om de inte är borttagna med **unset**, plus eventuella ändringar eller tillägg. Alla sådana måste markeras med kommandot **export**. Omgivningen av ett enkelt kommando kan ändras om det föregås av en eller flera parametersubstituioner. Sålunda är

```
TERM=twist kommando argument
```

och

```
(export TERM ; TERM=twist ; kommando argument)
```

likvärdiga (vad beträffar omgivningen för kommandot).

Om shelltillvalet **-k** sätts, placeras även de parametrar i omgivningen som definierats med strängar efter kommandot. Jämför kommandot *set* **-k**.

I första exemplet nedan skulle strängen TERM=twist behandlats som en parameter till kommandot, om inte tillvalet **-k** hade satts. I det andra exemplet, tar kommandot *set* **+k** bort tillvalet **-k**.

```
set -k
kommando TERM=twist argument
set +k
echo TERM=twist behandlas som en vanlig sträng!
```

SIGNALER

Signalerna INTERRUPT och QUIT ignoreras av ett startat kommandot om kommandot följs av &; annars har signalerna de värden som shell ärvde av sin förälder, dock inte signal 11 (se kommandot *trap* nedan).

BEARBETNING AV SHELLKOMMANDON

Varje gång ett kommando utförs (exekveras) , görs ovan nämnda substitutioner. En ny process skapas (utom när det gäller specialkommandon och definierade användarfunktioner, se nedan) och ett försök att starta kommandot görs via *exec*. Specialkommandon och funktioner utförs inom nuvarande shell.

Shellparametern PATH anger sökvägen (pathname) i filsystemet till de bibliotek som innehåller kommandon. Alternativa biblioteksnamn skiljs med kolon (:). Standard sökväg är **:/bin:/usr/bin** . Detta anger aktuellt bibliotek, /bin och /usr/bin i den ordningen). Lägg märke till att aktuellt bibliotek anges som ett tomt namn, som kan placeras omedelbart efter likhetstecknet eller mellan kolonseparatorer var som helst i PATH. Om kommandonamnet börjar med /, används inte PATH och kommandot kan inte utföras med en restriktiv shell. Annars genomsöks alla bibliotek i PATH efter en exekverbar fil. Biblioteken söks i den ordning de definierats i parametern PATH. Om filen får exekveras (användas som ett kommando) men inte är en **a.out** fil, antas den innehålla shellkommandon. Ett subshell (dvs en separat process) startas för att utföra dessa. Även en kommandolista inom parenteser bearbetas i en subshell.

Shell kommer ihåg sökvägen för varje kommando som utförts. Detta snabbar upp bearbetningen då samma kommando används flera gånger. De lagrade sökvägarna beräknas på nytt om nödvändigt, t ex tas alla relativa sökvägar bort när det aktuella biblioteket ändras (kommandot *cd*), för att lagras på nytt nästa gång de används. Specialkommandot *hash* används för att visa eller ta bort sökvägar som lagrats i shell. Om kommandot *hash* **-r** anges eller om PATH-variabeln ändras, tas alla lagrade sökvägar bort.

SPECIALKOMMANDON

Följande kommandon utförs inom shellprocessen och ingen omdirigering av in/utdata tillåts om så inte särskilt anges.

:

Ingen bearbetning. Alla parametrar till och med ; eller radslut ignoreras och utgångsstatus är noll. Läsning och tydning av specialtecken i parametrarna utförs därmot liksom all substitution av parametrar.

#

Ingen effekt. Hela resterande rad ignoreras och anses vara en kommentar.

. fil

Kommandon från fil läses och bearbetas. Den definierade sökvägen i PATH används för att hitta det bibliotek som innehåller filen. Observera! Ett mellanslag måste finnas mellan punkten och filnamnet. På detta sätt kan även ett kommando i en fil med samma namn som ett specialkommando

eller en definierad funktion (internt i shell) bearbetas, utan att filnamnet specificeras med ett helt pathname.

break [n]

Lämnar *for*- eller *while*-loop om sådan genomlöps. Är n angivet bryter kommandot n nivåer.

continue [n]

Återupptar nästa iteration i den genomlöpta *for*- eller *while*-loopen. Om n anges återupptas bearbetningen i den n:te loopen.

cd [newdir]

Ändrar aktuellt bibliotek till newdir. Som standard används parametern HOME som nytt bibliotek. Shellparametern CDPATH, om definierad, ger sökvägen för newdir om newdir inte börjar med tecknet /. Utan CDPATH söks newdir enbart i det aktuella biblioteket. Kommandot *cd* får inte utföras från en restriktiv shell.

echo [arg ...]

Ekar argument till standard output. Se separat beskrivning av kommandot *echo*.

eval [arg ...]

Argumenten utvärderas och utförs som kommandon. Detta möjliggör en andra tolkning av metatecken, efter en kommandosubstituering under den första utvärderingen. Se separat beskrivning av kommandot *eval*.

exec [arg ...]

Det kommando som angivits som argument till *exec* ersätter nuvarande shell och startas. Kommandot *exec* utförs utan att en ny process skapas. Om *exec* ges i en shell-procedur, bearbetas inte de kommandon i denna script som följer efter *exec*. Omdirigering av in- och utdata tillåts. Om inga argument ges påverkar eventuell omdirigering nuvarande shell, dvs en fil kan öppnas med andra filbeskrivare än standard 0,1,2 och standard filbeskrivare kan omdirigeras. Se även separat beskrivning av kommandot *exec*.

exit [n]

Avslutar shell med utgångsstatus n. n är ett värde mellan 0 och 255. Standardvärde för n är utgångsstatus från det senast utförda kommandot. Shell kan också avslutas av ett end-of-filetecken (normalt CTRL-D) som anges istället för ett kommando. Se även separat beskrivning av kommandot *exit*.

export [namn . . .]

Shell-parametrar med specificerade namn markeras för automatisk export till omgivningen av alla kommandon som utförs från nuvarande shell. Om inget argument anges, skrivs en lista ut med alla parametrar som exporteras. Se även separat beskrivning av kommandot *export*.

getopts

Används i shell-scriptar för att stödja kommando-syntax standard, det avkodar positionsparametrar och kollar efter tillåtna tillval. Se även separat beskrivning av kommandot *getopts*.

hash [-r] [namn . . .]

Med tillvalet -r tar detta kommando bort alla sparade sökvägar i shell. Med bifogade argument söker shell och sparar sökvägarna till dessa kommandon. Om inga argument ges, skrivs en lista ut över alla sparade sökvägar, tillsammans med statistisk information, kallad 'cost' och 'hits'. Cost ökas med 1 varje gång en ny sökning måste göras och visar det arbete som krävs för att lokalisera kommandot. Hits anger det antal gånger ett kommando har utförts. En asterisk (*) ges efter hits för kommandon vars sökvägar måste omvärderas.

newgrp [namn . . .]

Likvärdig med *exec newgrp* arg ....., dvs kommandot *newgrp* utförs utan att en ny shell skapas. Se en separat beskrivning av *newgrp*.

pwd

Skriver ut aktuellt bibliotek. Se beskrivning för kommandot *pwd* för detaljer.

read [namn . . .]

En rad läses från standard input och det första ordet tilldelas parametern, angivet som första argument, det andra tilldelas andra parametern osv. Överblivna ord tilldelas det sista parametern. Utgångsstatus är noll under förutsättning att ingen end-of-file påträffas.

readonly [namn . . .]

De givna namnen märks som readonly och dess värde kan inte ändras av efterföljande tilldelningar. Om inget argument anges, visas istället en lista med alla readonly-namn. Se också separat beskrivning av kommandot *readonly*.

return [n]

Avslutar en intern funktion med utgångsstatus n. Standard utgångsstatus är utgångsstatusen från det senast utförda kommandot inom funktionen.

set [[+][-]aefhknotuvx-] [arg ...]

Kommandot *set* visar eller ändrar aktuell omgivning för shell. Shelltillval, variabler och/eller positionsparametrar kan omdefinieras. Observera att shelltillvalen -r, -c, -s och -i inte kan ändras. Se även separat beskrivning av kommandot *set*.

Argument, om sådana finns, omdefinierar positionsparametrarna $1, $2, $3,... som är satta när shell startas. Utan tillval eller argument skriver kommandot *set* ut alla definierade shellvariabler. Om tillvalen ges med ett plus tecken (+) istället för det vanliga bindestrecket (-) framför tillvalsbokstaven, tas motsvarande tillval bort. Aktuell uppsättning av shelltillval finns i den speciella shellparametern $-, vilken kan ses med *echo* $-.

-a	Markerar för export alla modifierade eller skapade variabler.
-e	Avslutar omedelbart när kommandots utgångsstatus inte är lika med noll. Har ingen effekt om shell är interaktivt.
-f	Inga filnamn skapas. Tecknen *, ? and [...] tolkas inte för att skapa filnamn.
-h	Söker och kommer ihåg sökvägarna till kommandona i funktioner när funktionerna definieras. Normalt lokaliseras funktionskommandona först när funktionerna utförs.
-k	Alla shell parameterdefinitioner på en kommandorad placeras i kommandots omgivning, inte endast de som kommer före kommandot.
-n	Läser kommandon men utför dem inte. Använd inte detta tillval i interaktiv mod.
-t	Avslutar aktuell shell efter att ha läst och utfört nästa kommando. Bara avsett att användas för C-program; skall inte användas interaktivt.
-u	Behandlar icke definierade variabler som fel vid substitueringar. Normalt behandlas en odefinierad parameter som en tom sträng.
-v	Skriver ut rader som läses av shell allteftersom de läses.
-x	Skriver ut kommandon och deras argument allteftersom de utförs.
--	Detta är ingen tillval. Det är ett argument som avslutar tillvalen och möjliggör att följande argument kan starta med bindestreck (-) utan att därför tolkas som tillval.

shift [n]

Positionsparametrarna flyttas n steg och döps om. $(n+1) $(n+2) .... döps om till $1 $2 .... Standardskift är ett steg om inte n ges. Då blir $2 $3 ... ändrat till $1 $2 ...

test

Utvärderar villkorliga uttryck. Se separat beskrivning av kommandot *test*.

times

Skriver ut ackumulerad användar- och systemtid för processer startade från shell. Endast avslutade kommandon medräknas. Se separat beskrivning av kommandot *times*.

trap [cmd] [n] . . .

Kommandot *cmd* kommer att läsas och utföras när shell får signalen (signalerna) n. Ovbservera att *cmd* läses en gång när det definieras och en

gång när det utförs. *trap*-kommandon utförs i signalnummerordning. Det högsta tillåtna signalnumret är 16. Varje försök att sätta *trap* på en signal som ignorerades då shell startades är utan verkan. Alla försök att fånga signal 11 (memory fault) ger fel. Om argumentet *cmd* saknas kommer kommandot att återställa alla *trap* för de givna signalerna (n) till sina standardvärden. Om *cmd* är en nollsträng, t ex ", kommer denna signal att ignoreras av shell såväl som av de kommandon startas. Om signalen noll (0) specificeras utförs kommandot när aktuell shell avslutas. Kommandot *trap* utan argument skriver ut en lista på alla kommandon med tillhörande signalnummer. Jämför kommandot *kill*.

type [name . . .]

Shell avkodar varje argument som ett kommando och skriver ut hur argumentet tolkas. Hela sökvägen (pathname) skrivs ut för kommandon funna i en fil, medan strängen 'is a shell built-in' skrivs ut för kommandon som är interna i shell. Se även en separat beskrivning av kommandot *type*.

ulimit [-f] [n]

Kommandot *ulimit* sätter/skriver ut en max storlek för filer eller pipe-buffertar, som gäller för nya processer som startas. Vid läsning används ingen begränsning. Utan tillval är tillval **-f** standard. Utan argument skrivs nuvarande gräns ut.

-f Filstorleken begränsas till n block (512 bytes/block).

umask [ooo]

Användarens mask för filtillstånd vid skapande av nya filer sätts till ooo, där o är en oktal siffra. Om ooo utelämnas skrivs maskens aktuella värde ut. För ytterliga detaljer se separat beskrivning av kommandot *umask*. Se även kommandot *chmod*.

unset [namn . . .]

Shellparametrarna givna som argument tas bort och görs odefinierade. Parmetrarna PATH, PS1, PS2, IFS och MAILCHECK kan inte tas bort. Se även separat beskrivning av kommandot *unset*.

wait [n]

Inväntar avslutning av alla underprocesser eller väntar på en viss specificerad process med det givna processnumret n. Vid invänandet av en specificerad process blir utgångsstatus från *wait* den inväntade processens utgångsstatus. Utan argument inväntas alla underprocesser och utgångsstatus blir noll (0). Se även beskrivningen av kommandot *wait*.

START AV SHELL OCH TILLVAL

Om shell är en login shell, dvs startas med *exec*, och det första tecknet i argument 0 är -, läses kommandon först från **/etc/profile** och sedan från **$HOME/.profile**, om dessa filer existerar. Därefter läses kommandon enligt nedan, vilket också görs om shell startas som ett kommando (*/bin/sh*). Tillvalen nedan tolkas bara av shell vid starten; lägg märke till att om inte tillvalen **-c** eller **-s** satts, tolkas det första argumentet som ett namn på en fil innehållande kommandon, och resterande argument skickas till denna kommandofil som positionsparametrar:

Följande tillval kan endast sättas när shell startas. Likaså kan alla tillvalen beskrivna under specialkommandot *set* ovan ges när shell startas.

-V	Skriver ut versionsnumret för kommandot *sh*.
-c *sträng*	Om tillval **-c** är satt läses kommandon från strängen.
-s	Om tillval **-s** är satt eller om inga argument återstår, läses kommandon från standard input. Kvarvarande argument används som positionsparametrar. Utdata från shell, utom från specialkommandon, skrivs till standard error output (filbeskrivare 2).
-i	Shell är interaktivt om tillval **-i** är satt eller om in- och utdata för shell är anslutna till en terminal. I så fall ignoreras signalen TERMINATE (så att kommandot *kill* 0 inte ska stoppa en interaktiv shell) och signalen INTERRUPT fångas och ignoreras (så att *wait* kan avbrytas). Shell ignorerar alltid QUIT-signalen.
-r	Om tillval **-r** är satt, är shell en restriktiv shell. Se nästa avsnitt.

RESTRIKTIV (Begränsad) SHELL

Shell startas som en restriktiv shell antingen med tillval **-r** eller när specialvariabeln SHELL innehåller bokstaven r eller när shell startas med namnet *rsh*, länkat till */bin/sh*. Jämförd med en fullständig shell får användaren inte göra följande:

- Ändra aktuellt bibliotek (*cd*).
- Ändra variabeln PATH.
- Specificera filnamn eller kommandonamn innehållande /.
- Omdirigera in- eller utdata (> eller >>).

Dock, när kommandot *rsh* startas som en login-shell bearbetas filerna **/etc/profile** och **.profile** med fullt tillstånd innan begränsningarna ovan aktiveras. På detta sätt kan en användare skapas med speciellt begränsade tillstånd, eftersom **.profile** kan innehålla automatiska kommandon för att sätta upp variablerna PATH och HOME och för att flytta användaren till ett lämpligt bibliotek.

Kommandon i kommandofiler bearbetas med en ny normal shell, om inte tillvalet **-r** anges. På detta sätt kan en användare med en restriktiv shell utföra ett begränsat antal tillåtna kommandon med full styrka hos shell. Fullt skydd kräver att läs- och exekveringstillstånden för de tillgängliga biblioteken sätts upp för att förhindra att egna exekverbara kommandofiler skapas.

TILLVAL

Beskrivna i avsnittet START AV SHELL OCH TILLVAL.

FILER

/etc/profile
$HOME/.profile
/tmp/sh*
/dev/null

HÄNVISNING

cd(1), chmod(1), echo(1), eval(1), exec(1), exit(1), export(1), login(1), machid(1), pwd(1), readonly(1), set(1), test(1), times(1), umask(1), unset(1), wait(1)

FELMEDDELANDEN

Utgångsstatus från shell är satt till icke-noll när fel upptäcks, t ex syntaxfel. Om shell utför ett kommandoflöde, dvs ej är interaktivt då ett fel upptäckt, avbryts bearbetningen av kommandofilen. Annars returneras utgångsstatuset från det senast utförda kommandot. (Se kommandot *exit* ovan).

ANMÄRKNINGAR

Kommandot *readonly* (utan argument) ger samma lista som kommandot *export.*

Om ett kommando utförs och ett kommando med samma namn installeras i ett biblitek i sökvägen enligt PATH, före det biblioteket där ursprungliga kommandot påträffades, blir inte sökvägen som sparats i shell automatiskt ändrad och shell fortsätter att använda det ursprungliga kommandot. Använd kommandot *hash* **-r** för att rätta till detta.

Om man flyttar det aktuella biblioteket (kommandot *mvdir*) eller dess förälderbibliotek, kan kommandot *pwd* ge fel resultat. Använd då kommandot *cd* med fullständiga sökvägen (pathname) för att rätta till denna situation.

SH(1)

NAMN

shutdown - avslutar alla processer

SYNTAX

/etc/shutdown [-V] [-k] [tid] [Meddelande ..])

FUNKTION

Kommandot *shutdown* används av super-user för att på ett ordnat och försiktigt sätt avsluta alla pågående processer. Endast super-user kan använda kommandot och endast från den fysiska huvudkonsolen **/dev/console**.

Utan tillval, avslutar *shutdown* alla processer och går ner i enanvändarmod (run-level 's'). *shutdown* skall användas istället för kommandot *telinit* för att försiktigt stänga av ett system.

Med tillvalet **-k**, stängs systemet av helt. Normalt återstartas och laddas det igen (bootas) om inte nyckeln på framsidan vrids till läget OFF (eller motsvarande, beroende på maskinvaran) omedelbart efter att avstängningen är helt klar, dvs när följande text visas på skärmen.

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

Argumentet tid anger tiden i maximalt antal minuter innan systemprocesserna avslutas. Detta för att ge inloggade användare möjlighet att rensa och logga ut innan systemet stängs av. Standard-tiden är 5 minuter. Det sista argumentet är en valfri meddelande-sträng som ges till alla användare för att informera om anledningen till systemets avstängning.

shutdown går igenom följande steg: Först ges ett systemmeddelande tillsammans med meddelandeargumentet, vari alla inloggade användare uppmanas logga ut. Sedan uppdateras och stänges alla mountade filsystem innan systemet stängs av. Detta måste ske innan systemet återstartas för att hålla filsystemet rent.

TILLVAL

-V	Skriver versionsnumret för kommandot *shutdown*.
-k	Total avstängning av systemet.

FILER

Inga

HÄNVISNING

sync(1M), telinit(1M), powerfail(1M)

FELMEDDELANDEN

Ett felmedelande visas och ingen avstängning sker om *shutdown* startas från en annan terminal än den fysiska huvudkonsolen (**/dev/console**).

ANMÄRKNING

Det finns tre normala sätt att stänga av systemet. Detaljerna beror på vilket datorsystem som används, men i huvudsak gäller följande:

- För att gå ner i enanvändarmod, ges kommandot *shutdown* av superuser från huvud-konsolen.
- Om systemet skall återstartas utan att slå av strömmen, används kommandot *shutdown* **-k**. Inom 10 sekunder efter att meddelandet System halted visats, kan man vrida nyckeln till OFF/MAN för att initiera en manuell start. Om nyckeln lämnas i positionen AUTO får man en normal automatisk återstart av systemet.
- Det normala sättet att stänga av systemet helt och slå av strömmen varierar för olika datorsystem. Detta beskrivs i aktuell **Systemadministration**. I vissa system initieras avstängningen automatiskt genom att vrida nyckeln på framsidan, medan andra system kräver att systemadministratören använder kommandot *shutdown* **-k** innan strömmen slås av.

NAMN

siv - skärmorienterad text editor

SYNTAX

siv [-Vrxcs] [-a *sträng*] [filnamn]

FUNKTION

siv är en skärmorienterad text editor som kan hantera filer av alla storlekar.

siv hanterar filer med icke-ASCII-tecken genom att visa alla kontrolltecken som \nnn där nnn är ett tresiffrigt oktalt värde. Genom att använda terminaler med utökade teckenuppsättningar kan *siv* hantera 8-bitarstecken som skrivbara tecken.

Vid start, kan *siv* konfigureras med användarkommandon och/eller ge teckenkonvertering genom att läsa en initieringssekvens från filen **/etc/.siv**. Dessutom finns möjlighet att med tillvalet **-a** ange en sträng med *siv*-kommandon som automatiskt skall utföras. En användare kan ha sin egen **.siv** fil i HOME-biblioteket. Shell-variabeln TERM måste definieras och motsvarande parameterfil måste finnas i biblioteket **/usr/lib/terminfo** för att *siv* skall kunna användas.

TILLVAL

-V	Skriver versionsnumret för kommandot *siv*.
-r	Begränsar *siv*. Tillåter inte exekvering av shellkommandon.
-x	Begränsar tangentbordet. XON/XOFF tangenterna (CTRL-S/CTRL-Q) används inte som *siv* kommandon. Istället används CTRL-] och CTRL-\.
-c	När man startar *siv* med detta tillval och utan att ange ett filnamn, används det sist använda filnamnet, vilket sparats i filen **.siv_file** vid sista normala avslutningen av *siv* (med CTRL-X CTRL-F). (Se **-s** tillvalet).
-s	Med tillvalet **-s** kommer *siv* att spara editerade filens filnamn i filen **.siv_file** i det aktuella biblioteket om *siv* avslutas normalt vid CTRL-X CTRL-F.
-a sträng	*siv*-kommandon i strängen utförs automatiskt när *siv* startas. Om strängen inte innehåller ett avslutningskommando, kan användaren fortsätta att editera därefter. Strängen skall omges med apostrofer och kontrolltecken kan anges som tecknet ^, följt av motsvarande bokstav (exempel: CTRL-S anges som ^S).

FILER

/etc/.siv	Standardinitiering
/usr/lib/terminfo/*/*	Terminalparametrar
$HOME/.siv	Användarens initiering
.siv_file	Spar senaste filnamn

EXEMPEL

I detta exempel, med tillvalet **-a**, blir alla förekomster av texten 'Firma HB' utbytta mot texten 'Firma AB' i filen.

```
siv -a '^[QFirma HB
Firma AB
!
^X^F' filen
```

HÄNVISNING

dmacs(1)

NAMN

sleep - uppskjuter bearbetningen

SYNTAX

sleep [-V] tid

FUNKTION

Kommandot *sleep* uppskjuter bearbetningen med det antal sekunder som är angivna med argumentet tid. Maximal fördröjning är 65535 sekunder (ungefär 18 timmar).

sleep används antingen för att utföra ett kommando efter en viss tid eller att utföra ett kommando upprepade gånger.

TILLVAL

-V Skriver ut versionsnumret för kommandot *sleep*.

FILER

Inga

EXEMPEL

Exempel 1:

```
sleep 105 ; cat filnamn
```

Kommandot *cat* utförs efter 105 sekunder.

Exempel 2:

```
while true
do
   echo 'command sleep'
   sleep 37
done
```

Kommandot *echo* utförs upprepade gånger. Fördröjningstiden är 37 sekunder för varje gång.

HÄNVISNING

sh(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

Den angivna tiden måste vara mindre än 65535 sekunder.

SLEEP(1)

NAMN

sort - sorterar och sammanlägger filer.

SYNTAX

sort [-VcmubdfinrtxM] [+pos1 [-pos2]] ...[-z *recsz*] [-y *kmem*] [-o *utfil*] [-s *sorttab*] [filer]

FUNKTION

Kommandot *sort* sammanlägger filer och sorterar raderna från alla namngivna filer och skriver därefter ut resultatet på standard output. Ett streck (-) som filbeskrivning i argumentet filer betyder att indata ska tas från standard input. Om inga filnamn anges kommer standard input att sorteras.

Sorteringen sker normalt på hela rader. Standard sorteringsordning är angiven i filen **/usr/etc/sorttable.** Om denna fil inte existerar sker sortering lexikografiskt efter bytevärdena enligt de principer som används i datorn, vanligen enligt ASCII-värdena. Tillvalet **-s** kan välja en annan fil som sorteringsordnings-fil. Sorteringsordningsfilen innehåller 256 sorterade ASCII tecken. Sorteringen påverkas globalt av de angivna kommando-tillvalen. Fält-tillval kan läggas till individuellt i varje definition av en sorteringsnyckel och upphäver alla globala tillval.

Beteckningen +pos1 and -pos2 begränsar nyckeln för sorteringen till det fält som börjar i pos1 och slutar före pos2. Om pos2 saknas betyder det slutet på raden. Både pos1 och pos2 har formen **m.n** och de kan följas av en eller flera av fälttillvalen **bdfinrM**. I **m.n** är värdet **m** en siffra som anger hur många fält som hoppas över från början av raden och **n** anger hur många ytterligare tecken som ska hoppas över.

+m.n betyder att sorteringsnyckeln startar vid n+1:sta tecknet i det m+1:sta fältet på raden. Om **.n** saknas betyder det **.0**, dvs det första tecknet i m+1:sta fältet. Om tillvalet **-b** är angiven räknas **.n** från första icke blanka tecknet i detta fält.

-m.n betyder att sorteringsnyckeln slutar med n:te tecknet, separatorer inräknade, efter m:te fältet. Om **.n** saknas betyder det **.0**, dvs till och med sista tecknet i det m:te fältet. Exempel: För att använda hela första fältet som sorteringsnyckel används de två argumenten +0 -1.

Om några fälttillval angetts upphäver dessa alla globala sorteringstillval för denna nyckel. Fälttillvalen läggs till -pos2 oberoende av +pos1. I **-tx** tillvalet är fälten strängar, separerade med **x** (dvs tecknet som anges efter **-t**). Annars är fälten icke-tomma-strängar separerade med ett blanktecken. I definitionen av en sorteringsnyckel inkluderas även blanktecken som följer fält-separatorn.

Om det finns flera sorteringsnycklar kommer de senare nycklarna att användas först när de tidigare nycklar gett en sann jämförelse. Rader som är likvärdiga sorteras med alla tecken signifikanta.

TILLVAL

-V Skriver ut versionsnumret för kommandot *sort*.

-b Hoppar över inledande blanktecken (mellanslag och TAB) vid start- och slutpositioner i en sorteringnyckel. Ett globalt **-b** tillval gäller för alla +pos1 och -pos2 argument. Annars kopplas **-b** fälttillvalet till +pos1 och -pos2 självständigt. Observera att tillvalet **-b** är effektivt endast när begränsade sorteringsnycklar specificerats.

-d Sorterar i "Lexikon"-ordning, dvs bara bokstäver, siffror och mellanslag har betydelse.

-f Översätter versala bokstäver till gemena under jämförelsen. Filen **/usr/etc/sorttable.f** innehåller översättningstabell. Om denna fil inte existerar konverteras endast a-z. En annan fil kan användas om tillvalet **-s** anges.

-i Ignorerar tecken utanför det oktala ASCII området 040 - 0176 i icke-numeriska jämförelser.

-n En numerisk sträng bestående av blanktecken, minustecken, en nolla eller siffror med eventuell decimalpunkt kommer att sorteras efter det aritmetiska värdet. Tillvalet **-n** ger automatiskt tillvalet **-b**, om någon begränsad sorteringsnyckel används.

-r Vänder sorteringsordningen.

-t*x* Tecknet *x* separerar fälten istället för, som standard, mellanslag och/eller tabbar. Varje befintligt *x* är betydelsefullt, t ex *xx* avgränsar ett tomt fält.

-c Kontroller att indatafilen är sorterad enligt sorteringsreglerna. Inget data skrivs ut så vitt inte filen är osorterad, i vilket fall ett felmeddelande ges och utgångsstatus från kommandot *sort* blir skilt från noll.

-m Slår endast samman filerna. Infilerna är redan sorterade.

-u Tar bort alla kopior av en rad. Ignorerade tecken och tecken utanför sorteringsnycklarna ingår inte i denna jämförelse.

-o *utfil* *utfil* är namnet på den utfil som ska användas istället för standard output. Filen kan vara samma fil som någon av infilerna.

-s *sorttab* Väljer en annan tabellfil än standard för sorteringsordning. Standardnamnet är **/usr/etc/sorttable**. Med tillvalet **-f**, används filnamnet (med tillägget **.f**) även för konvertering mellan versaler och gemena.

-M Jämför strängar sorterade som månader på ett år, dvs JAN < FEB < MAR < ... Stora och små bokstäver är likställda i denna jämförelse och endast de tre första icke-blanka tecknen i ett fält jämföres. Ogiltiga fält anses vara < JAN. Tillvalet **-b** tillämpas. Månadsnamnen är på engelska.

-y [*kmem*] Argumentet *kmem* anger storleken i kbyte av det interna minne som ska användas istället för standard när *sort* startas. Med detta tillval sorteras stora filer snabbare eftersom *sort* inte behöver begära ytterligare minne varje gång ett minnesblock krävs. **-y0** tvingar *sort* att börja med minsta möjliga utrymme och **-y**, utan argumentet *kmem*, tvingar *sort* att använda det största möjliga minne som systemet tillåter.

-z *recsz* Argumentet *recsz* är den maximalt tillåtna radlängden, i bytes. Normalt sätter *sort* den maximala radlängden till längsta raden i infilerna under sorteringsfasen, och allokerar lämpliga buffertareor. Med tillvalen **-c** eller **-m** genomgås emellertid ingen sorteringsfas och en standard buffert-storlek (512 bytes) används om inte tillvalet **-z** anges. I så fall avbryts *sort* med felmeddelandet om en längre rad än tillåten skulle läsas in. Detta förhindras med tillvalet **-z**.

FILER

/usr/tmp/stm???	Temporära filer.
/usr/etc/sorttable	Standard fil för sorteringsordning.
/usr/etc/sorttable.f	Standard teckentabell för konvertering till gemena (-f tillvalet).

EXEMPEL

Exempel 1:

```
sort -u +0f +0 list
```

Skriver ut i alfabetisk ordning alla unika ord i filen **list**, som innehåller ett ord per rad. Ord med stora respektive små bokstäver tolkas som olika ord.

Exempel 2:

```
l | sort -r +4
```

I detta exempel sorteras fillistan från kommandot **l**, enligt 5:e fältet, som anger filstorleken. Sorteringen är omvänd och listar den största filen först.

Exempel 3:

```
sort -t: +2n /etc/passwd
```

Skriver ut lösenordsfilen sorterad numeriskt enligt användar-ID (tredje fältet) fälten separeras med kolon ":".

Exempel 4:

```
sort -uM +0 -1 dates
```

Skriver ut första händelsen i varje månad från en redan sorterad fil med månad och dag som första två fält på raderna. Med tillvalen **-u** och **-M** och bara en infil kommer säkert just första påträffade av en följd likadana rader att skrivas ut.

HÄNVISNING

uniq(1), mksort(1)

FELMEDDELANDEN

Ges av olika anledningar. Vid fel blir utgångsstatus satt till noll. Om inte sista raden avslutats med newline tillfogar *sort* en sådan men skriver ut ett varningsmeddelande till standard error.

ANMÄRKNING

I den vanliga sorteringsordningsfilen (**/usr/etc/sorttable**) sorteras de svenska tecknen åääö och ÅÄÖ enligt svensk standard. Kommandot *mksort* kan användas för att skapa en ny sorteringsordningsfil.

NAMN

stty - sätter/visar terminalparametrar för seriella portar.

SYNTAX

stty [-V] [-a] [-g] [moder]

FUNKTION

Kommandot *stty* sätter eller skriver ut vissa kommunikationsparametrar (modes = lägen) för den enhet som är aktuell standard input. När en parameter anges som argument ändras bara denna parameter.

Utan argument meddelar *stty* hur parametrarna är satta. Utan tillval visas endast de viktigaste parametrarna, om möjligt med användade av kombinationsläge (se nedan). Med tillvalen **-a** eller **-g** visas alla parametrar.

Argumentet moder (lägen) kan vara i mnemonisk form eller numerisk form.

Det finns fem grupper av parametrar: styrlägen, input-lägen, output-lägen, lokala lägen, och kontrolltecken-bestämningar. Dessutom kan några kombinationslägen användas för att ange vanliga kombinationer av normala parametrar.

I beskrivningen nedan har många lägen ett TILL-läge och ett FRÅN-läge. FRÅN-state väljs/visas genom att ett minustecken (-) anges före lägesnamnet.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *stty*.
-a	Alla parametrar skrivs ut.
-g	Alla parametrar skrivs ut som en rad med numeriska fält ett fält per parametergrupp, som kan användas som argument till ett annat *stty* kommando.

STYRLÄGEN

parenb (-parenb) Sätter generering av paritet för utdata och kontroll av paritet för indata TILL (FRÅN). Paritetskontroll av indata kan hindras med läget **-inpck**.

parodd (-parodd) Sätter udda (jämn) paritet. Ignoreras såvida inte **parenb** är satt.

cs5 cs6 cs7 cs8 Sätter tecken-storlek i bitar, exklusive paritetsbitar.

0 Gör omedelbar hangup ("Lägger på luren"). Används temporärt för att tvinga fram nerkoppling av linjen. Se även **hupcl** nedan.

50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 19200 38400 exta extb

Sätter, om möjligt, överföringshastigheten till angivet värde (baudrate).

hupcl (-hupcl) Kopplar ner linjen (kopplar inte ner) vid sista close. Hung-up betyder att utsignalen DTR släpps i den seriella kontakten för att tvinga fram bortkoppling av ett modem från linjen.

hup (-hup) Samma som **hupcl (-hupcl)**

cstopb (-cstopb) Använd två (en) stoppbit(ar) per tecken.

cread (-cread) Sätter på (stänger av) mottagaren.

clocal (-clocal) Förutsätter en linje utan (med) modemkontroll.

INPUT-LÄGEN

ignbrk (-ignbrk) **Observera:** Ignorerad i D-NIX. Break-tecken ignoreras alltid (ignbrk).

brkint (-brkint) Genererar (genererar inte) signalen SIGINT vid BREAK i indata. Ignoreras om **ignbrk** är satt.

ignpar (-ignpar) Ignorerar (ignorerar inte) tecken i indata med paritets- eller framing-fel.

parmrk (-parmrk)

Konverterar (konverterar inte) indatatecken med paritets- eller 'framing'-fel. Detta läge ignoreras om **ignpar** eller **-inpck** är satta. Ett fel översätts till en tre-tecken-sekvens, 0377, 0, X, där X är mottagen byte. Om **-istrip** är satt, konverteras ett inläst verkligt tecken med värdet 0377 till två tecken (0377,0377). Vid **-parmrk**, konverteras istället ett fel till en NULL byte (ASCII 0).

inpck (-inpck) Sätter på (stänger av) paritetskontroll av indata. Ignoreras såvida inte läget **parenb** är satt. Vid **-inpck** kombinerad med **parenb** kan paritet genereras för utdata utan paritetskontroll av indata.

istrip (-istrip) Förkortar (förkortar inte) tecken i indata till 7 bitar. Detta görs efter eventuell paritetskontroll.

inlcr (-inlcr) Konverterar (konverterar inte) NL till CR i indata.

igncr (-igncr) Ignorerar (ignorerar inte) CR i indata.

icrnl (-icrnl) Konverterar (konverterar inte) CR till NL i indata. Detta läge ignoreras om **igncr** är satt.

iuclc (-iuclc) Konverterar (konverterar inte) versala bokstäver (A-Z) till gemena (a - z). Andra tecken i indata ändras inte.

ixon (-ixon) Sätter på (stänger av) START/STOP styrning av utdata. Sändning av utdata stoppas vid mottagning av ASCII DC3 (CTRL-S) och startas igen vid mottagning av ASCII DC1 (CTRL-Q) från den externa enheten. Med **ixon** läses

aldrig tecknen av tecknen DC3 och DC1 som indata. Se även **ixany**.

ixany (-ixany) Tillåter alla mottagna tecken (endast DC1) att återstarta sändning av utdata. Se **ixon**.

ixoff (-ixoff) Begär att systemet sänder (inte sänder) STOP/START tecken när indatabufferten är nästan full/tom. DC3 sänds för att stoppa och DC1 för att tillåta mottagning av indata. Se **ixon**.

OUTPUT-LÄGEN

opost (-opost) Konverterar (konverterar inte) utdata. Med **(-opost)** ignoreras alla andra outputlägen och utdata sänds enligt styrlägena.

olcuc (-olcuc) Konverterar (konverterar inte) gemena bokstäver (a-z) till versala (A-Z) i utdata.

onlcr (-onlcr) Konverterar (konverterar inte) NL till CR-NL i utdata.

ocrnl (-ocrnl) Konverterar (konverterar inte) CR till NL i utdata.

onocr (-onocr) När **onocr** är satt, sänds inget CR tecken om kolumnpekaren redan är i kolumn 0.

onlret (-onlret) NL på terminalen antas (antas inte) utföra CR-funktionen lokalt på den externa enheten. CR fördröjning används och kolumnpekaren sätts till 0 vid NL i utdata. Ignoreras om **onlcr** är satt.

ofill (-ofill) Använder utfyllnadstecken (använder tidsfördröjning) för fördröjningar. Tidsfördröjning möjliggör längre fördröjningar än utfyllnadstecknen.

ofdel (-ofdel) Utfyllnadstecken är DEL (NUL).

nl0 nl1 Sätter fördröjning efter 'new-line' i utdata. Ignoreras om **onlret** är satt, i vilket fall en CR-fördröjning används istället. Notera att andra tecknet i läges-namnet är lilla bokstaven l. 'nl0' innebär ingen fördröjning. 'nl1' specificerar omkring 0.1 sekunds fördröjning (eller två utfyllnadstecken).

cr0 cr1 cr2 cr3 Väljer fördröjning efter CR i utdata.
cr0 innebär ingen fördröjning.
cr1 beror på positionen inom raden (eller 2 utfyllnadstecken)
cr2 innebär ca 0.1 sekunder (eller 4 utfyllnadstecken).
cr3 innebär ca 0.15 sekunder (Tidsfördröjning används alltid).

bs0 bs1 Väljer fördröjning efter back-space i utdata. bs0 innebär ingen fördröjning. bs1 innebär ca 0.05 sekunder (eller 1 utfyllnadstecken).

ff0 ff1 Väljer fördröjning efter form feed i utdata. ff0 innebär ingen fördröjning. ff1 innebär ca 2 sekunder (Tidsfördröjning används alltid).

vt0 vt1 Väljer fördröjning efter vertikal tabulering i utdata. vt0 innebär ingen fördröjning. vt1 innebär ca 2 sekunder (Tidsfördröjning används alltid).

tab0 tab1 tab2 tab3

Sätter expansion av TAB-tecken eller fördröjningstyp efter TAB i utdata. Ett TAB-tecken är ASCII-tecknet 011 oktalt.
tab innebär ingen fördröjning och ingen expansion.
tab1 innebär fördröjning beroende av radpositionen (eller 2 utfyllnadstecken).
tab2 innebär ca 0.10 sekunders fördröjning (eller 2 utfyllnadstecken).
tab3 specificerar expansion av TAB till strängar av mellanslag (spaces).

Kommentarer till fördröjningslägena vid utdata:

Fördröjningarna ovan är ungefärliga och är beroende av linjehastigheten och systembelastningar. Inga fördröjningar kan specificeras på TC.

LOKALA LÄGEN

isig (-isig) Avkodning av (inte avkodning av) kontrolltecknen INTR och QUIT i indata. Om bara ett av dessa skall kontrolleras kan man ange **isig**, men modifiera kontrolltecknets funktion till 'odefinierad'.

icanon (-icanon) Sätter på (stänger av) kanonisk konvertering av indata. Om kanonisk konvertering väljs, blir kontrolltecknen **erase** och **kill** avkodade och eventuellt aktiverad **xcase**-konvertering utförs. Dessutom kommer också tecknen NL, EOF och EOL att avsluta inläsning av indata. Om **-icanon** anges kan en inläsning ske snabbt blockvis. Varje indatablock måste innehålla minst **min** tecken eller avslutas efter en väntetid mellan tecknen (timeout) på **time**, innan blocket sänds iväg till den läsande processen. Om **min** är noll sänds blocket vid timeout även om det är tomt. Om värdet **time** är noll, används ingen timeout. Om både **min** och **time** är noll, sänds blocket direkt även om det är tomt. Värdena **min** och **time** definieras som kontrolltecken. Se nedan. Observera att **min** och **time** lagras i samma position som kontrolltecknen **eof** och **eol**, varför **-icanon, min** och **time** alltid bör sättas samtidigt. Av samma orsak bör **icanon, eof** och **eol** alltid återställas samtidigt.

xcase (-xcase) Kanonisk (oprocessad) överföring av versaler/gemener. Ignoreras om **-icanon** anges. Normalt används detta läge i kombination med **iuclc** och **olcuc**. Om **xcase** är

aktiv kan versala bokstäver läsas in genom att citera motsvarande gemena bokstav med tecknet \. Vissa specialtecken kan också läsas in som speciella sekvenser enligt tabellen nedan. Om dessa bokstäver och specialtecken skrivs ut i utdata konverteras de och blir automatiskt föregångna av tecknet \. Följande specialsekvenser avkodas i indata och genereras i utdata. Här antages **-iuclc** och **-olcuc** aktiva.

Tecken	Inlästes som:	Utskrivs som:
A	\a eller A eller \A	\A (Lika för A-Z)
a	a	a
\	\\	\\
|	| eller \!	\!
{	{ eller \(	\(
}	} eller \)	\)
~	~ eller \^	\^
`	` eller \'	\'

echo (-echo) Ekar (ekar inte) alla indatatecken till utdata.

echoe (-echoe) Om även **echo** är satt, ekas ERASE-tecken som strängen backspace-mellanslag-backspace. Detta raderar nyss inskrivet tecken på de flesta bildskärmsterminaler. Däremot håller den inte reda på kolumnpositioner och kan därför förvilla vid escapesekvenser, tab-tecken och tidigare backspace. Ignoreras om **-icanon** är satt. Om **-echo** är satt ekas ERASE tecken som strängen mellanslag-backspace.

echok(-echok) Ekar (ekar inte) NL efter inläsning av kontrolltecknet KILL. Ignoreras om **-icanon** är satt.

echonl (-echonl) Ekar (ekar inte) NL, även när **-echo** är satt. Lämplig när halv duplex används. Ignoreras om **-icanon** är satt.

noflsh (-noflsh) Aktiverar (förhindrar) tömning (flush) av in- och utdata buffertar efter inläsning av tecknen INTR och QUIT.

KONTROLLTECKEN-BESTÄMNINGAR

kontrolltecken C

Definierar *kontrolltecken* som C. Kontrolltecknet kan vara en av: **intr, quit, erase, kill, eof, eol** eller **swtch**. Om tecknet C är en kontrolltangent, dvs med ett ASCII-värde mellan 0 och 31, anges det genom tecknet ^ följt av motsvarande tangent. C ska då omges av apostrofer. Alternativt kan kontrolltangenten anges direkt om den citeras, dvs förgås av tecknet \. Exempelvis kan CTRL-H användas för att radera föregående tecken vid inläsning om det definieras genom kommandot **stty erase '^H'** . Speciellt avkodas '^?' som tecknet DEL och '^-' anger att kontrolltecknet är odefinierat. Kontrolltecknen avkodas vid normal inläsning

av data enligt generella regler i terminalhanteraren, modifierat av stty-parametrarna. I indata kan tecknen för **erase, kill** och **eof** citeras med \ för att de inte skall avkodas. Om **-icanon** är vald avkodas inte **erase, kill, eof** eller **eol** och värdena för **eof** och **eol** används som **min** respektive **time**-värden (se nedan).

Kontrolltecknen är:

intr	Tecken som ger signalen INTR (om **isig** angivits).
quit	Tecken som ger signalen QUIT (om **isig** angivits).
erase	Tecken som raderar föregående tecken. Vanligen är den '^H' eller '#'.
kill	Tecken som raderar hela raden.
eof	Tecken som anger slut på filen. Vanligen är denna CTRL-D, dvs '^D'.
eol	Tecken som anger slut på raden förutom standardtecknet LF. Vanligen används den inte och är satt till NULL, dvs '^@'.
swtch	Reserverad. Används för närvarande ej.

min nn (där 0 <= nn <= 255)

time nn (där 0 <= nn <= 255)

Dessa variabler används bara när **-icanon** är vald och anger då villkor för blockvis mottagning av indata enligt beskrivningen för **-icanon** ovan. Notera att dessa är synonymer för **eof** (=**min**) respektive **eol** (=**time**), men där värdena anges numeriskt. Eftersom **eof** och **eol** vanligen är definierade som CTRL-D och NULL är då **min**=4, **time**=0 om de inte ändras när **-icanon** sätts. Tiden anges i tiondels sekunder. Se **-icanon** ovan för beskrivning av dessa variabler.

line nn (där enbart nn lika med 0 används)

Sätter linjedisciplinen 0 < nn < 127. För närvarande är endast linjedisciplin 0 implementerad.

KOMBINATIONSLÄGEN

evenp eller parity

Sätter **parenb** och **cs7**.

oddp Sätter **parenb, cs7** och **parodd**.

-parity eller -evenp eller -oddp

Sätter **-parenb** och **cs8**.

raw (-raw) Sätter på (stänger av) rå in-/utmatning. Rå I/O innebär att ingen av kontrolltecknen ERASE, KILL, INTR, QUIT eller EOT avkodas och ingen konvertering av utdata utförs. Inga fördröjningar görs.

nl (-nl) nl sätter lägena: **-icrnl -onlcr**, **-nl** sätter lägena: **icrnl onlcr -inlcr -igncr -ocrnl -onlret** (Notera att andra tecknet i lägesnamnet **nl** är bokstaven l, inte siffran 1).

lcase (-lcase) Sätter (tar bort) **xcase, iuclc** och **olcuc**.

LCASE (-LCASE) Samma som **lcase (-lcase)**.

tabs (-tabs eller tab3)

Behåller (expanderar) TAB i utdata. **tabs** sätter tab0.

ek Sätter ERASE- och KILL-tecknen till de normala CTRL-H och CTRL-U.

sane Sätter alla lägen till rimliga standardvärden. Användbar när en terminal inte uppför sig som väntat. Baudrate förändras inte i detta läge. **Observera:** Ge kommandot *stty* med parametern **sane** som nedan för att undvika problem vid felaktig uppsättning av CR/NL konvertering: CTRL-J *stty* **sane** CTRL-J.

term Sätter alla lägen lämpade för terminal-typen ***term***, där ***term*** är någon av tty03, tty37, vt05, tn300, ti700 eller tek. De enda skillnaderna är fördröjningarna.

EXEMPEL

Exempel 1:

Sätt jämn paritet och 7 databitar, men ignorera alla paritetsfel i indata. Specificera kontrolltecknet kill till CTRL-U och maximal fördröjning vid radslut i utdata.

```
stty evenp istrip -inpck kill '^u'-ofill cr3 nl1
```

Exempel 2:

Detta är ett exempel på ett *stty*-kommando som kan användas i filen **.profile**:

```
stty -tabs cr0 ff0 nl0 erase '^H' kill '^X'
```

HÄNVISNING

Inga

FELMEDDELANDEN

Inga

ANMÄRKNING

Många kombinationer är meningslösa, men ingen kontroll utförs på oförenliga kombinationer i kommandot *stty*.

STTY(1)

NAMN

su - ändra användare-ID eller bli super-user

SYNTAX

su [-] [-V] [användarnamn [argument ....]]

FUNKTION

Med kommandot *su* kan man byta användar-ID utan utloggning. Operatören skall alltid ange lösenordet för den nya användaren på terminalens tangentbord, även om standard input omdirigeras. Förfrågan om lösenordet ges till standard error output.

Ett nytt användarnamn kan anges som parameter till *su*. Om inget namn anges blir användarnamnet 'root' och användaren loggas in som superuser. Utan ett specificerat användarnamn kan inga andra argument ges på kommandoraden för *su*.

su startar en ny shell, eller ett annat program, enligt det sista fältet för den nya användaren i filen **/etc/passwd**. Både den verkliga och den effektiva användaridentiteten sätts enligt den nya användaren. Tillval och kommandoargument som ges på kommandoraden efter användarnamnet överföres till nya shell eller det nya programmet.

Om det första argumentet till *su* är ett minustecken '-' får man samma resultat som om användaren loggade in. Både **/etc/profile** och **.profile** i det nya hembiblioteket bearbetas av nya shell (eller andra program enligt **/etc/passwd**).

Alla försök att bli ny användare med kommandot *su* loggas till filen **/usr/adm/sulog**.

Observera: Beskrivningen nedan gäller endast när en normal shell startas (**/bin/sh**).

För att återvända till den föregående användaridentiteten används end-of-file (CTRL-D) eller kommandot *exit*. Om shell kommandon ges som argument till *su* återfås den förra användaridentiteten automatiskt efter avslutning av dessa kommandon.

Shell-tillval med argument kan ges som argument till *su*. Shellkommandon kan ges för direkt utförande om de föregås av tillvalet **-c**. Flera kommandon kan specificeras i en sträng med ';' som separator om argumenten omges av tecknen " ".

Exempel:

```
su - kalle -c "pwd ; dmacs yourfile".
```

Exporterade shell variabler ändras normalt inte vid byte av användare, med följande undantag: a) när man blir super-user sätts alltid prompten PS1 och variabeln PATH till default # och **/bin:/etc:/usr/bin** och b) när tillvalet - ges till *su*, ändras hela miljön till standardmiljön för den nya användaren.

Observera att filen **.profile** kan testa kommandoargumentet $0 för *-sh* eller *-su* för att avgöra på vad sätt det har aktiverats. Den nya shell'en

kommer också att ha ett annat processnamn i listan erhållen av kommandot *ps*. *sh* är en original login shell, *sh* är en shell som anropas direkt med kommandot *sh*, är en shell som anropas med kommandot *su* och *su* är en shell som anropas med *su* -, dvs med argumentet -.

TILLVAL

-V Skriver ut versionsnumret för kommandot *su*.

FILER

/usr/adm/sulog	Loggning av kommandot su.
/etc/passwd	Filen med lösenord.
/etc/profile	Normala login kommandon
$HOME/.profile	Login-kommandon för den nye användaren

EXEMPEL

Exempel 1:

```
su Password:******
```

När kommandot *su* angivits, skrivs en förfrågan ut på standard error för lösenordet till super-user (root). När rätt lösenord angivits, skriver systemet vanligen ut prompten för super-user, '#'. När användaren har utfört sina uppgifter som super-user kan återgång till normal användaridentitet ske med end-of-file (CTRL-D).

Exempel 2:

```
su - kalle
Password:*****
```

Användaren vill logga in som 'kalle'. Detta är möjligt om användaren känner till kalles lösenord. Med tillvalet - kommer hela miljön att sättas upp som vid normal inloggning som användare Kalle. När arbetet utförts för 'kalle' loggar man ut till normalanvändaren med CTRL-D.

Exempel 3:

```
su root -c "dmacs /etc/timezone"
Password:*****
```

Systemet frågar efter lösenordet för super-user och editorn *dmacs* startas för att ändra filen **/etc/timezone**. När editorn *dmacs* avslutas sker återgång till normalanvändare automatiskt.

HÄNVISNING

sh(1)
/etc/passwd
.profile

FELMEDDELANDEN

Felmeddelanden visas om det angivna användarnamnet inte finns i filen **/etc/passwd**.

ANMÄRKNING

Notera att 'aktuellt bibliotek' ej ingår i PATH efter byte till superuser (root).

NAMN

sync - uppdaterar filsystem.

SYNTAX

sync [-V]

FUNKTION

Kommandot *sync* tömmer buffrad information till yttre skivminnesenheter. Detta sker oavsett om buffrarna är fulla eller ej. Kommandot kan användas för att tvinga ut data i filsystemen. Att kommandot *sync* ges garanterar emellertid inte att informationen skrivs ut till en fysisk enhet, även om en kontroll görs. *sync*-kommandot avslutas nämligen så snart skrivningen har påbörjats.

Kommandot måste användas innan systemet stängs av, vilket sker automatiskt med normala avstängningsrutiner.

TILLVAL

-V Skriver ut versionsnumret för kommandot *sync*.

HÄNVISNING

shutdown(1M), mkcfig(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

Att man använt kommandot *sync* garanterar inte att informationen verkligen har skrivits till den yttre enheten.

Systemet aktiverar automatiskt *sync*-funktionen internt med jämna mellanrum, vanligen en gång per minut. Se *mkcfig*.

Det finns en situation när *sync* **inte** skall användas. Det är efter modifiering av det fysiska filsystemet med kommandot *fsck*. Därför får kommandot *fsck* aldrig användas för att modifiera ett mountat filsystem.

SYNC(1M)

NAMN

tar (tape archive) - sparar/återställer filer i bandarkiv och andra media.

SYNTAX

tar [-]nyckel [nyckelargument] [filnamn ....]

FUNKTION

Kommandot *tar* sparar och återställer filer på magnetband eller andra media såsom disketter. Filerna sparas i ett speciellt format (*tar*-format) med titelblock vilka ger en tillförlitlig kontroll av biblioteksstrukturen, filtyperna, åtkomsttillstånd och andra parametrar. Detta möjliggör för ett senare *tar*-kommando att läsa och rekonstruera hela eller delar av sparade filer. Titelblock i arkivet är skyddade genom checksummor. Det sparas dock inte någon checksumma för filinnehållet, endast för filstorleken.

Arkivet kan också sparas till eller återställas från en fil eller standard input/output. Se tillvalet **-f** nedan.

Vad kommandot *tar* exakt gör bestäms helt av första argumentet, nyckeln, vilken kan föregås av ett minustecken (-). Nyckeln består av en sträng med minst en funktionsbokstav och normalt en eller flera funktionsmodifierare.

Övriga argument specificerar fil/filer eller bibliotek som skall sparas eller återställas. Om ett bibliotek angivits, blir det sparat/återställt med alla sina underbibliotek och filer. Observera att alla metatecken (wildcards) som används i argumenten blir avkodade av shell med hjälp av biblioteken i det aktuella filsystemet och inte från filnamnen på arkivet.

TILLVAL

Tillvalen består av två delar, en **funktionsbokstav** (r, x, t, u, c), och **funktionsmodifierare** (v, w, f, b, l, m, o, 0, ... ,7 k, n, F, V, A, S, s, y).

Modifierarna k, n, ... y är tillägg till standarden för kommandot *tar*.

Funktionsbokstav

r Sparar de angivna filerna i slutet av band- eller diskettarkivet.

x Läser de angivna filerna från arkivet och återställer dem i filsystemet. Om ett angivet filnamn är ett bibliotek som har sparats, blir detta bibliotek hämtat rekursivt. Om inte funktionen ändras med funktionsmodifierarna **m**, **o** eller **A**, gäller följande:

Ägare, grupp och modifieringstid återställs från arkivet. ID för ägare och grupp är numeriska på arkivet och kan motsvara olika namn i olika system. Åtkomsttillstånden återställs endast för filer som inte redan finns i filsystemet. Dock återställs tillståndsmoderna 'set-user-ID' och 'set-group-ID' endast om *tar* används av super-user.

Om inga filnamn anges, hämtas hela arkivets innehåll.

TAR(1)

Observera: Om det finns flera kopior av samma fil i arkivet, skriver sista kopian över alla föregående när filen återställs. Detta möjliggör uppdatering av ett befintligt arkiv genom att lägga till nya versioner av filerna efter de äldre versionerna. Detta görs med funktionsbokstaven **u**.

t Listar filerna i arkivet till standard output. Filerna listas varje gång de påträffas i arkivet. Om inga filnamn är angivna, listas hela arkivet. Med modifieraren **v**, ges en komplett lista med mer information.

u Sparar de specificerade filerna i slutet av arkivet, men bara om de inte redan finns där eller om de blivit modifierade sedan de sist sparades på arkivet.

c Sparar de angivna filerna i ett nytt arkiv. Inskrivning startar i början av bandet/disketten istället för efter sista filen.

Funktionsmodifierare

v Normalt arbetar *tar* i det tysta. Om modifieraren **v** sätts, listas varje fil i listan. I listan föregås filnamnet av funktionsbokstaven i *tar*. Om funktionsbokstaven **t** också sätts, listas all filinformation från arkivet i en form som liknar den för kommandot *ls* **-l**. Listningen sker till standard output.

w Skriver ut filnamn och vilken åtgärd som skall vidtas och låter användaren konfirmera interaktivt. Om svaret börjar med etty, utförs åtgärden, annars inte.

f Kommandot *tar* använder nästa argument som namn på arkivet istället för standardnamnet **/dev/mt1**. Denna funktion krävs alltid för åtkomst till disketter eller streamerkassetter. Om endast ett minustecken (-) är angivet som nästa argument skriver *tar* arkivet till standard output eller läser det från standard input, beroende på funktionsbokstaven. Detta gör det möjligt att använda *tar* som start och slut i en filterkedja. Kommandot *tar* kan på detta sätt användas för att flytta hierarkier, t.ex:

```
cd fromdir ; tar -cf - . | ( cd todir ; tar -xf - )
```

b När skrivning sker på ett rått bandarkiv eller standard output, använder kommandot *tar* det följande argumentet (decimaltal) som blockfaktor, dvs. antalet block i varje sektion av bandet. Ett *tar*block är alltid 512 bytes. Standard-blockfaktor är 1 och max värdet är 20.

När arkivet är på andra externa enheter, såsom disketter, är den verkliga blockfaktorn alltid 20 och **b**-modifieraren påverkar endast buffertstorleken vid överföringen.

Om en blockfaktor större än 20 anges, använder kommandot *tar* den maximala blockfaktorn (20), och värdet som givits som argument kommer istället att ange bufferstorleken under dataöverföringen. Ju större bufferten är, desto fortare sker överföringen. När arkiv läses (funktionstangenterna **x** eller **t**), sätts blockfaktorn automatiskt från arkivet och eventuell **b**-modifierare ignoreras.

l Kommandot *tar* rapporterar om det inte kan lösa upp alla länkar till filer sparade på ett arkiv. Om denna modifierare inte har angivits, skrivs inte några felmeddelanden ut.

m Kommandot *tar* kommer inte att återställa modifieringstiden. Istället för modifieringstiden kommer tiden då hämtningen gjordes att anges som modifieringstid för de åter-ställda filerna.

o Denna modifierare ger de hämtade filerna samma användar- och grupp ID:s som användaren av kommandot *tar* har.

0..7 Ett tal mellan 0 .. 7 anger den externa enheten. Default är 1.

Följande modifierare har lagts till i D-NIX.

k Kommandot *tar* kommer att tolka nästa argument som storleken på arkivvolymen i kbyte. Minsta tillåtna värde är 250 och storleken måste vara en multipel av blockstorleken i kbyte (tillval **-b**). Denna modifierare bör användas om storleken på en volym, t.ex ett band eller en diskett är begränsad. Väldigt stora filer blir delade och sparas på flera volymer. När dessa skapas och återskapas kommer kommandot *tar* att fråga efter en ny volym så länge som en fil inte är komplett.

n Anger att arkivenheten inte är ett band. Modifieraren **k** är underförstådd. Listning och hämtning av ett arkiv snabbas upp eftersom *tar* kan hoppa över filer under sökning.

F Kommandot *tar* använder nästa argument som namn på en fil från vilken resten av kommandoraden tas. Innehållet i denna fil läggs till den givna kommandoraden. På detta sättet kan många filer specificeras för att spara eller återställa.

V *tar* - programmets versionsnummer skrivs ut i formatet: *tar* Rn.nn.

A När en fil återställs från ett arkiv, kommer eventuella inledande / , vilket anger rootbiblioteket, att tas bort från filnamnen. Detta gör det möjligt att återställa filer

till ett temporärt bibliotek även om de sparades med fullständiga pathnamn.

S Gör det möjligt att också spara specialfiler (vanligen i biblioteket **/dev**) i ett arkiv. Av kompabilitetsskäl är detta dock inte standard-funktion i *tar*. Denna modifierare behövs inte när det gäller att återställa filer från ett arkiv. Kommandot *tar* kan emellertid aldrig återställa specialfiler som redan finns i ett filsystem. De existerande specialfilerna som skall ersättas måste tas bort med kommandot *rm* **-i** före inläsningen av de nya versionerna från arkivet.

s Med denna modifierare, använder *tar* en snabbare intern överföring vid läsning från arkivet (funktionsbokstav **x** eller **t**). Modifieraren **s** får inte användas med funktionsbokstäverna **r** eller **u** eller med media genererat av en annan dator.

y Detta är en speciell modifierare, som används vid läsning på ett media där fel har upptäckts. Modifieraren gör det möjligt att återställa filer från en position på mediat efter det upptäckta felet. Med hjälp av modifieraren **y**, söker *tar* efter titelblock utan att först spola tillbaka bandet.

För att nå bästa resultat, bör block faktor 1 specificeras. Det är tillrådligt att stegvis kopiera bandarkivet till en diskfil, från vilken *tar* då kan läsa med blockfaktorn satt till ett. Kopieringen kan göras med kommandot *dd*. Exempel:

```
dd if=/dev/nst0 of=/tmp/sttmp bs=100k
```

och *tar*:

```
tar -xvfby /tmp/sttmp 1
```

FILER

/tmp/tar*

EXEMPEL

Exempel 1:

```
tar -cf /dev/mf0 *
```

Alla filer i det aktuella biblioteket kommer att sparas på en 5 1/4 tums diskett. Modifieraren **c** talar om för *tar* att det är ett nytt arkiv som skall skapas. Modifieraren **f** talar om för *tar* att nästa argument i kommandosträngen (**/dev/mf0**) anger enhetsnamnet (eller filen) där arkivet skall placeras. Tecknet '*' anger att alla filer skall sparas, utom de som börjar med en punkt, .

Exempel 2:

```
tar -tf /dev/mf0
```

Med dessa modifierare läser kommandot arkivet och skriver ut en lista på filerna och biblioteken samt deras storlek. Blockfaktorn skrivs också ut.

Exempel 3:

```
tar tvf /dev/mf0
```

Funktionen **t** anger listning av filerna som finns i arkivet. Argumentet (**/dev/mf0**) för modifieraren **f**, är namnet på arkivet, här en 5 1/4 tums diskett. Om modifieraren **v** också används, blir listningen mer fullständig. Filtyp, åtkomsttillstånd, ägare, grupp, storlek och all annan information för varje fil listas.

Exempel 4:

```
tar -xf /dev/mf0 kalle
```

Kommandot läser från arkivet på **/dev/mf0** och återskapar filen eller biblioteket med namnet kalle.

Exempel 5:

```
tar -cvf /dev/mf0 usr/myfile
```

Funktionsbokstaven **c** anger att ett nytt arkiv skapas, **v** betyder att tar listar det som händer på standard output, **f** talar om att nästa argument (**/dev/mf0**) är arkivet på vilket biblioteket eller filen **usr/myfile** skall sparas.

Exempel 6:

```
tar -xvf /dev/mf0 myfile
```

Läsning från arkivet indikeras med **x**. Argumentet till **f** är arkivnamnet. **v** listar vad som sker och kommandot *tar* återställer filen eller biblioteket **myfile** från arkivet. Hela arkivet läses om inget bibliotek/fil anges. Observera att inga "wildcards" får användas i filnamnet. För att återställa filen sparad i exempel 5 ovan, skall namnet **usr/myfile** anges.

Exempel 7:

```
tar -cvfk /dev/mf0 720 *
```

Kommandot skapar ett nytt arkiv på en 720 kbytes diskett. Modifieraren **k** med argumentet 720, är storleken på arkivvolymen i kbytes. Modifieraren **k** anger också att när/om arkivvolymen blir full skall *tar* begära av användaren att sätta in en ny diskett. Det är viktigt att argumentens ordning följer funktionsmodifierarnas ordning.

```
tar -cvfk /dev/mf0 720 *
```

är lika med:

```
tar -cvkf 720 /dev/mf0 *
```

Exempel 8:

```
tar -rvf /dev/mf0 myfile
```

Biblioteket eller filen **myfile** kommer att läggas till i slutet av arkivet, även om den redan finns där och inte har ändrats.

Exempel 9:

```
tar -uvf /dev/mf0 myfile
```

Biblioteket eller filen **myfile** kommer att läggas till i slutet av arkivet, men **endast** om den saknas eller blivit ändrat. Inga filer tas bort från arkivet med detta kommandot, dvs den gamla **myfile** finns kvar i arkivet men kopieras över, när sista filen **myfile** i arkivet läses.

Exempel 10:

```
cd /
tar -cvfbk /dev/st0 200 60300 usr
```

Hela biblioteket **/usr** på systemet sparas på bandstreamern för att gälla som backup. Blockfaktor 20 används, men modifieraren **b** och dess argument 200 indikerar att en stor I/O-buffert (100 kb) används för att skynda på överföringen. En varning kommer att skrivas ut eftersom den givna blockfaktorn är större än 20, men kan i detta fall ignoreras. Bandstorleken antas vara 60 kbytes, med en rimlig marginal.

Exempel 11:

```
tar -cvf/dev/mf0 .
```

Hela biblioteks-strukturen från det aktuella biblioteket kopieras på en diskett. Observera att då *tar* söker biblioteken rekursivt är detta ett passande sätt att inkludera samtliga filer, inklusive filer som börjar med .

HÄNVISNING

bup(1), copy(1), cpio(1), dd(1M)

FELMEDDELANDEN

Felmeddelanden visas på standard error output om:

- Om felaktig tillvalsnyckel angivits.
- Om ett läs/skriv-fel påträffats på en fysisk enhet.
- Om minnet är otillräckligt för en länktabell.

ANMÄRKNING

Det är inte möjligt att på något sätt fråga efter n:te förekomst av en fil medan arkivet läses. Band-fel behandlas mycket primitivt. Filnamn får inte vara längre än 100 tecken.

Åtkomsttillstånden för en redan befintlig biblioteksfil återställs från arkivet bara om användaren har super-user privilegier.

Återställda ägare- och gruppnamn kan skilja sig från originalnamnen om filerna sparats med en annan dator där andra ID-nummer använts.

Funktionsbokstäverna **r** och **u** för *tar* är hårdvaruberoende. De kan t.ex. i allmänhet inte användas tillsammans med den vanliga bandstreamerenheten såvida denna inte medger att läsning kan ske utan återspolning. Tidigare versioner av *tar* i D-NIX än version 1.07, hade funktionsmodifieraren **o** som standard.

NAMN

tc - Övervakningsprogram för terminalkoncentrator

SYNTAX

/etc/tc [-V] [-s tid]

FUNKTION

Detta är ett start- och övervakningsprogram för alla terminalkoncentratorkort (TC) i systemet. Programmet laddar initialt ner programvara till alla TC-kort och startar upp dem. Det övervakar sedan TC-korten med valbart tidsintervall. Om något TC-kort inte svarar, laddas programvaran åter till kortet och återstartas. Övervakningsprogrammet skapar också alla enheter om dessa inte redan existerar.

Kommandot */etc/tc* startas normalt automatiskt när systemet startas.

OPTIONER

-V	Skriver ut versionsnumret för kommandot *tc*.
-s *tid*	*tid* anger (i sekunder) hur ofta */etc/tc* ska testa om TC-korten fortfarande är aktiva. Standard är 15 sekunder.

FILER

/dev/tty*nn* Terminalportar, *nn*=siffror.
/dev/tc*n* Enhet för varje TC-kort *n*=siffra.

HÄNVISNING

init(1M)

TC(1M)

NAMN

telinit - styr verksamheten av processen init

SYNTAX

telinit [-V] [0123456sqabch]

FUNKTION

Telinit, som är länkad till filen **/etc/init**, används för att styra verksamheten i processen *init*. Den tar ett argument som är ett tecken och signalerar *init* via systemanropet kill till att uföra en lämplig handling. Argumentet fungerar som direktiv till *init (telinit)*. Kommandot telinit kan endast användas av super-user eller av en medlem i gruppen sys.

Funktionerna beskrivs mer detaljerat under kommandot *init*.

0-6	instruerar *init* att placera systemet i ett av systemnivåerna 0-6.
a,b,c	instruerar *init* att starta de processer som i **/etc/inittab** har markerats med systemnivåerna a, b eller c.
q	instruerar *init* att åter gå igenom filen **/etc/inittab**.
s	instruerar *init* att övergå till enanvändarmod. När denna nivåändring har skett ändras den virtuella systemkonsolen, **/dev/syscon**, till den terminal från vilken kommandot utförts. **Observera:** Normalt skall kommandot *shutdown* användas vid övergång från fleranvändarmod till enanvändarmod. Se *shutdown*.
h	instruerar *init* att omedelbart stänga av systemet och stoppa processorn. Filsystemet stängs av korrekt. **Observera:** Använd detta kommando enbart vid fel. Normalt stoppas systemet med nyckeln eller med */etc/shutdown* **-k**.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *telinit*.

FILER

/etc/inittab
/dev/syscon

HÄNVISNING

init(1M), shutdown(1M), who(1)

FELMEDDELANDE

Inga

ANMÄRKNING

telinit bör **inte** användas för att övergå från fleranvändarmod till enanvändarmod. Istället skall *shutdown* användas.

Kommandot *who* **-r** kan användas för att lista nuvarande systemnivå.

NAMN

test - testar villkorsuttryck

SYNTAX

test uttryck

eller

[uttryck]

FUNKTION

test utvärderar uttrycket och returnerar en nolla som utgångstatus om uttrycket är sant. Utgångsstatus skilt från noll ges om argument saknas eller om uttrycket returnerar en tom sträng. Kommandot *test* används ofta tillsammans med satsen **if** i shellprocedurer.

Kommandot *test* är ett internkommando i shell och exekveras utan att det skapas någon ny process. Omdirigering av in/utdata förekommer inte.

TILLVAL

Följande uttryck kan användas för att konstruera villkoret:

-r ***file***	Sann om filen existerar och är läsbar.
-w ***file***	Sann om filen existerar och är skrivbar.
-x ***file***	Sann om filen existerar och är exekverbar.
-f ***file***	Sann om filen existerar och är en vanlig fil.
-d ***file***	Sann om filen existerar och är ett bibliotek
-c ***file***	Sann om filen existerar och är en teckenorienterad specialfil (enhet).
-b ***file***	Sann om filen existerar och är en blockorienterad specialfil (enhet).
-p ***file***	Sann om filen existerar och är en namngiven pipe (FIFO-fil).
-u ***file***	Sann om filen existerar och dess 'set-user-ID'-bit är satt.
-g ***file***	Sann om filen existerar och dess 'set-group-ID'-bit är satt.
-k ***file***	Sann om filen existerar och dess "sticky bit" är satt.
-s ***file***	Sann om filen existerar och har en storlek större än noll.
-t (*filbes*)	Sann om den öppna fil, vars filbeskrivarnummer är *filbes* (standardvärde är 1, dvs standard output), är ansluten till en terminalport.
-z ***s1***	Sann om strängen *s1* är definierad, men har längden noll.
-n ***s1***	Sann om strängen *s1* har en längd skild från noll.

s1 = *s2*	Sann om strängarna *s1* och *s2* är identiska.
s1 != *s2*	Sann om strängarna *s1* och *s2* inte är identiska.
s1	Sann om *s1* inte är en tom sträng.
n1 **-eq** *n2*	Sann om heltalet *n1* är lika med *n2*.
n1 **-ne** *n2*	Sann om heltalet *n1* inte är lika med *n2*.
n1 **-gt** *n2*	Sann om heltalet *n1* är större än *n2*.
n1 **-ge** *n2*	Sann om heltalet *n1* är större än eller lika med *n2*.
n1 **-lt** *n2*	Sann om heltalet *n1* är mindre än n2.
n1 **-le** *n2*	Sann om heltalet *n1* är mindre än eller lika med *n2*.
	Dessa sex jämförelser är tillåtna endast när både *n1* och *n2* är heltal och de jämförs algebraiskt.

Ovanstående uttryck kan kombineras med följande operatorer:

!	Negationsoperator.
-a	Binär OCH-operator.
-o	Binär ELLER-operator (**-a** har högre rang än **-o**).
\(**uttryck** \)	Parenteser, citerade från shell, kan användas för gruppering.

Observera att alla operatorer och flaggor måste separeras med mellanslag på kommandoraden, eftersom de är separata argument till kommandot *test*. Lägg också märke till att parenteser har betydelse för shell och således måste citeras med \.

En shellvariabel som kan vara odefinierad eller anta värdet 'tom sträng' måste omges av citationstecken "..." vid jämförelse, för att inte förväxlas med 'ingen sträng alls'. T.ex. kommandot *test* **-z "$A"** är alltid korrekt. Kommandot *test* **-z $A** däremot, kommer att tas som syntaxfel, om **$A** är en tom sträng.

FILER

Inga

EXEMPEL

Följande exempel visar hur en shellprocedur läser en rad från standard input (ett svar från operatören) och väljer att gå ur om användaren svarar j eller J.

```
echo 'Vill du gå ur (j eller n)?\c'
read A
if test "$A" = 'j' -o "$A" = 'J'
then exit
fi
```

HÄNVISNING

find(1), sh(1)

FELMEDDELANDEN

Felmeddelanden ges vid syntaxfel.

ANMÄRKNING

Om andra syntaxversionen används (dvs. den som använder [...] istället för ordet *test*), måste [och] vara åtskilda av blanktecken

TEST(1)

NAMN

time - exekveringstid för ett kommando.

SYNTAX

time [-V] kommando

FUNKTION

Kommandot *time* skriver ut den tid som det tar för det specificerade kommandot att utföras. Tre tider skrivs ut, den första anger den totala tiden för kommandot, den andra anger tiden då kommando-koden utfördes och den tredje anger systemtiden som kommandot tog i anspråk.

Tiderna anges i sekunder och informationen skrivs ut på standard error output när kommandot avslutats.

Exekveringstiden beror på typen av minne som programmet finns i.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *time*.

FILER

Inga

EXEMPEL

Exempel 1:

```
time ps
```

Kommandot *time* anger tiden för exekvering av kommandot *ps*.

Exempel 2:

```
time who who-file
```

Kommandot *time* anger den tid det tar för kommandot *who* att läsa filen **who-file** och skriva ut resultatet.

Exempel 3:

```
time ls -l
- lista från ls -
real     8.0
user     0.5
sys      2.3
```

Exemplet beskriver hur man får reda på hur lång tid det tar att utföra kommandot *ls* **-l**.

HÄNVISNING

ps(1)

FELMEDDELANDEN

Inga

TIME(1)

NAMN

times - skriver ut användarens ackumulerade systemtid.

SYNTAX

times

FUNKTION

Kommandot *times* skriver ut användarens ackumulerade systemtid för processer körda i nuvarande shell. Detta innehåller bara de totala tiderna för alla processer som har avslutats.

Tiderna anges som i exemplet nedan i minuter och sekunder.

```
0m44s 2m55s
```

Den första tiden anger den ackumulerade användartiden och den andra anger systemtiden.

Kommandot *times* är internt i shell och exekveras utan att en ny process skapas.

HÄNVISNING

sh(1)

NAMN

touch - uppdatera modifieringstid och användningstid för filer.

SYNTAX

touch [-Vamc] [mmddhhmm[yy]] filnamn ...

FUNKTION

Kommandot *touch* uppdaterar modifierings- och användningstiden av filerna genom att läsa ett tecken ur filen och sedan skriva tillbaka det igen. Både läs- och skrivtillstånd krävs för filerna.

Formatet för datum och tid är som beskrivs för kommandot *date*, där minst timme och minut måste anges. Om ingen tid anges, används nuvarande datum och tid.

```
mm =    månad ( 1 .. 12). Standard är nuvarande månad.
dd =    datum (1 .. 31). Standard är nuvarande datum.
hh =    timme. 24-timmarsklocka.
mm =    minut (0 .. 59).
yy =    år, sista två siffrorna i året. Standard är nuvarande år.
```

Om filen inte existerar görs ett försök att skapa den.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *touch*.
-c	Om filen inte existerar, görs inget försök att skapa den.
-a	Uppdaterar endast användningstiden.
-m	Uppdaterar endast modifieringstiden. Standard är annars att uppdatera båda, dvs. som om **-am** var angivet.

FILER

Inga

EXEMPEL

Exempel 1:

```
touch kap1
```

Modifierings- och användningstiden för filen kap1 uppdateras.

Exempel 2:

```
touch k02int.f k12int.f k22int.f
```

Modifierings- och användningstiden för filerna k02int.f, k12inf.f och k22int.f uppdateras.

HÄNVISNING

date(1)

FELMEDDELANDEN

Utgångsstatus från *touch* är det antal filer vars tider inte kunnat ändras plus det antal filer som inte fanns och inte skapades.

NAMN

tr - översätter tecken.

SYNTAX

tr [-Vcds] [sträng1] [sträng2]

FUNKTION

Kommandot *tr* kopierar standard input till standard output med substitution eller borttagning av utvalda tecken. Indata som finns i sträng1 ersätts av motsvarande tecken (i samma position) i sträng2. Alla kombinationer av **-cds** tillvalen är tillåtna.

Följande konventioner kan användas för att ange en följd tecken eller upprepade tecken i strängarna:

[a-z]	Anger en sorterad sträng av tecken med ASCII koder i området a till z, inklusive a och z.
[a*n]	Anger en sträng med n upprepningar av tecknet a. Om första siffran i n är noll, tolkas n som ett oktalt tal; annars tolkas n som ett decimaltal. Om n är 0 eller saknas helt, antas n vara ett mycket stort tal; denna funktion är användbar för utfyllnad av sträng2.

Citattecknet \ kan användas i shell för att ta bort specialbetydelsen av något tecken i strängen. Dessutom, betyder ett \ följt av en, två eller tre oktala siffror, det ASCII-tecken som har det angivna värdet.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *tr*.
-c	Komplementerar teckenuppsättningen i sträng1 med avseende på de tecken som har ASCII-koderna 001 till 377 oktalt, dvs. alla tecken utom de som är givna i sträng1 kommer att substitueras.
-d	Tar bort alla inkommande tecken som finns i sträng1.
-s	Komprimerar alla upprepade tecken i utdata, som finns i sträng2, till ett tecken.

FILER

Inga

EXEMPEL

Exempel 1:

Följande kommandorad skapar en lista av orden i fil1, och skriver ett ord till fil2. Ett ord är den största sträng som består av alfabetiska tecken. Strängarna är satta inom citationstecken "...." för att inte speciella tecken skall tolkas av shell; \012 är ASCII-koden för newline (ny-rad).

```
tr -cs "[A-Z][a-z]" "[\012*]" <fil1 >fil2
```

HÄNVISNING

sh(1)

FELMEDDELANDEN

Inga

ANMÄRKNING

tr accepterar inte ASCII NUL i sträng1 eller sträng2. NUL tas alltid bort från indata.

NAMN

true - ger utgångsstatus noll.

SYNTAX

true

FUNKTION

Kommandot *true* gör inget annat än att ge ett utgångsstatus noll. Det motsatta kommandot, *false*, gör inget annat än att ge ett utgångsstatus skilt från noll. *true* används vanligtvis i shellprocedurer som:

```
while true
do
   command
done
```

TILLVAL

Inga

FILER

Inga

HÄNVISNING

sh(1), false(1)

FELMEDDELANDEN

true ger alltid ett utgångsstatus (noll) som motsvarar ett kommando som avslutats korrekt utan fel.

TRUE(1)

NAMN

tty - ger terminalens namn

SYNTAX

tty [-V] [-s] [-l]

FUNKTION

Kommandot *tty* skriver ut pathname för användarens terminal på standard output. Tillvalet **-s** undertrycker utskrifter, så att endast utgångsstatus ges.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *tty*.
-s	Undertrycker utskrift.
-l	Skriver ut det synkrona linjenumret till vilket terminalen är ansluten.

FILER

Inga

HÄNVISNING

logname(1)

FELMEDDELANDEN

Utgångsstatus från *tty* är 0 om standard input är en terminal, annars 1. Utgångsstatus är 2 om ett ogiltigt tillval använts.

ANMÄRKNING

"not a tty" skrivs på standard output om tillvalet **-s** inte givits och om standard input inte är en terminal.

TTY(1)

TYPE(1)

NAMN

type - visar hela sökvägen för ett kommando

SYNTAX

type kommandonamn ... ...

FUNKTION

För varje kommandonamn skriver kommandot *type* ut ett helt pathname (sökväg) där motsvarande kommandofil hittats efter sökning enligt variabeln $PATH. Om kommandot är internt i shell skrivs strängen 'is a shell builtin' ut. Om det givna kommandonamnet inte är ett kommando skrivs strängen 'not found' ut. Utskrift sker till standard output.

Kommandot *type* är självt internt i shell och utförs utan att en ny process skapas. Omdirigering av utdata kan dock anges.

TILLVAL

Inga

EXEMPEL

Exempel 1:

```
type ls
```

Visar följande rad:

```
ls is /bin/ls
```

Exempel 2:

```
type cd
```

visar följande rad:

```
cd is a shell builtin
```

Exempel 3:

```
type mycmd
```

Visar följande rad om kommandot finns i användarens eget bin-bibliotek och om en av sökströngarna i $PATH är **/usr/kalle/bin**.

```
mycmd is /usr/kalle/bin/mycmd
```

HÄNVISNING

sh(1)

FELMEDDELANDE

Inga

TYPE(1)

NAMN

umask - definiera tillståndsmask för skapande av filer/bibliotek.

SYNTAX

umask [onum]

FUNKTION

Kommandot *umask* sätter umask-värdet. Utan argument visar den det aktuella värdet.

Detta umask-värde är ett tresiffrigt oktalt värde och anger de filtillståndsbitar som **inte** skall sättas när ett bibliotek eller en fil skapas. Se kommandot *chmod* för en beskrivning av åtkomsttillstånden. Vanliga kommandon skapar dessutom vanligen filer utan exekveringstillstånd (x).

Kommandot *umask* är internt i shell och utförs utan att en process skapas.

TILLVAL

Inga

EXEMPEL

```
umask 022
mkdir newdir
cat <oldfile >newfile
ls -l
drwxr-xr-x 2 kalle other 48 Dec 3 12:35 newdir
-rw-r--r-- 1 kalle other 4566 Dec 3 12:35 newfile
```

Detta umask-värde hindrar skrivtillstånd för 'gruppen' och 'alla övriga'. *mkdir* skapar därför biblioteket **newdir** med tillstånden 755 oktalt och kommandot *cat* skapar filen **newfile** i detta bibliotek med filtillståndet 644 oktalt. Resultatet kan läsas med kommandot *ls* **-l**.

HÄNVISNING

sh(1), chmod(1)

UMASK(1)

NAMN

umount - kopplar bort filsystem.

SYNTAX

/etc/umount [-V] enhet

FUNKTION

Kommandot *umount* informerar operativsystemets kärna att det förut mountade filsystemet ska stängas. Alla existerande systembuffertar skrivs ut till filsystemet och kopplingen till rootfilsystemet tas bort. Detta måste göras innan någon extern enhet tas bort fysiskt, t ex en diskett. Ett filsystem kan inte tas bort om någon fil däri är öppen eller om något bibliotek däri är aktuellt för någon användare.

Ytterligare detaljer beskrivs under kommandot *mount*. Kommandot *umount* är en länk till samma fil som kommandot *mount*.

TILLVAL

-V Skriver ut versionsnumret för kommandot *umount*.

FILER

/etc/mnttab
/etc/mtab

EXEMPEL

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

Kopplar bort ett tidigare filsystem på en 5 1/4 tums diskett.

HÄNVISNING

mount(1M), mntchk(1M)

FELMEDDELANDEN

Följande utgångsstatus erhålls då kommandot *umount* avslutas.

0 *umount* utfördes korrekt.

1 *umount* kunde ej utföras korrekt.

UMOUNT(1)

NAMN

uname - skriv ut namnet på det aktiva operativsystemet.

SYNTAX

uname [-Vsnrvmaduo]

FUNKTION

Kommandot *uname* skriver ut systemnamnet på operativsystemet till standard output. Kommandot används vanligen för att undersöka vilket system man använder. Informationen innehåller detaljer om både maskinvaran och programvaran.

TILLVAL

-V	Skriver ut versionen för kommandot *uname*.
-s	Skriver ut operativsystemets namn (detta är standard om inget tillval anges).
-n	Skriver ut nodnamn. Nodnamnet kan vara det namn som systemet är känt under för ett kommunikationsnät. Namnet definieras med kommandot *mkcfig*.
-r	Skriver ut releasenummer för operativsystemet.
-v	Skriver ut versionsnummer för operativsystemet.
-m	Skriver ut vilken maskinvara som används.
-a	Skriver ut all ovanstående information i ordningsföljden, **-s**, **-n**, **-r**, **-v**, **-m**.
-d	Skriver ut namnet på OEM-distributören.
-u	Skriver ut maskinvarans serienummer.
-o	Skriver ut tillverkarens namn.

FILER

Inga

HÄNVISNING

mkcfig(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

I D-NIX versioner före 5.2, användes **-m** tillvalet för att ange tillverkarens namn (nu tillval **-o**).

UNAME(1)

NAMN

uniq - rapportera upprepade rader i en fil

SYNTAX

uniq [-Vudc [+*n*] [-*n*] [infil] [utfil]]

FUNKTION

uniq läser infilen och jämför intilliggande rader. I normala fall tas den andra och därefter följande kopior av upprepade rader bort och den resterande filen skrivs till utfil. Filerna infil och utfil skall alltid vara olika filer. Om inga filer anges används standard input och output.

Observera att endast intilliggande upprepade rader behandlas med detta kommando. Använd annars kommandot *sort*.

TILLVAL

-V	Skriver ut versionsnumret för kommandot *uniq*.
-u	Anger att endast de rader som inte upprepas skrivs till utfil.
-d	Anger att en kopia av de upprepade raderna ska skrivas till utfil.
-c	Genererar rapport i standard format men varje rad föregås av ett tal som talar om hur många gånger raden fanns i filen. Tillvalen **-u** och **-d** ignoreras.
-n	De första *n* fälten samt tillhörande blanktecken ska hoppas över när man jämför rader. *n* är ett decimalt tal. Ett fält definieras som en sträng av tecken separerade av mellanslag och TAB-tecken från varandra.
+n	De första *n* tecknen hoppas över. *n* är ett decimalt tal. Fält hoppas över (med tillvalen *-n*) innan tecknen för tillvalet *+n* börjar räknas.

FILER

Inga

HÄNVISNING

sort(1)

FELMEDDELANDEN

Inga

UNIQ(1)

NAMN

unset - tar bort definitioner för shellvariabler

SYNTAX

unset namn ... ...

FUNKTION

Kommandot *unset* tar bort shellvariabler eller shellfunktioner angivna som argument. Efter borttagning anses de vara odefinierade.

Kommandot *unset* är internt i shell och utförs utan att en ny process skapas. Omdirigering av in/utdata är inte tillåtet.

Variablerna PATH, PS1, PS2, IFS och MAILCHECK kan inte tas bort om de är definierade.

TILLVAL

Inga

HÄNVISNING

sh(1), set(1)

UNSET(1)

NAMN

wait - väntar på att processen skall avslutas

SYNTAX

wait [nn]

FUNKTION

Kommandot *wait* inväntar terminering av alla processer som startats upp i bakgrunden (startade med &) från nuvarande shell och returnerar utgångsstatus 0. Om ett process-nummer är angivet som argument, blir endast denna process inväntad och *wait* får utgångsstatus enligt den processens utgångsstatus.

Eftersom systemanropet *wait* måste utföras av förälderprocessen, utförs det av shell självt, utan att någon ny process skapas.

Observera: Alla pipelines (innehållande kommandon med pipe-tecknet |) med tre eller fler steg anses inte vara barn till shell och kan därför inte väntas på, om de inte är särskilt angivna som argument till *wait*.

TILLVAL

Inga

FILER

Inga

HÄNVISNING

sh(1), ps(1)

FELMEDDELANDEN

Inga

NAMN

wall - skriver ut meddelande till alla inloggade användare, eller alla användare i en grupp.

SYNTAX

/etc/wall [-V] [-g group]

FUNKTION

Kommandot *wall* läser från standard input till end-of-file. Därefter sänds meddelandet till alla inloggade användare eller, med tillvalet **-g**, till alla användare i den angivna gruppen.

Meddelandet föregås av texten "Broadcast Message..." som talar om att det är ett meddelande till alla användare. Sändaren bör vara super-user för att gå förbi alla de skydd som varje enskild användare eventuellt kan ha satt upp. Jämför kommandot *mesg*.

TILLVAL

-V	Skriver ut versionen av kommandot *wall*.
-g	Ett meddelande sänds till alla användare i en grupp som specficerats som nästa argument. 'Grupp'-argumentet är ett namn och måste finnas i filen **/etc/group**. Meddelandet sänds till alla användare vilka är specificerade som användarnamn för gruppen i filen **/etc/group**.

FILER

/etc/group fil som innehåller gruppnamn och gruppmedlemsnamn.
/etc/utmp fil som innehåller alla inloggade användare.

EXEMPEL

Exempel 1:

```
/etc/wall
Systemet kommer att tas ned fredagen 24/11 kl 12.00
```

Detta meddelande som är skrivet på terminalen kommer att skickas ut till alla användare. Det kan se ut på följande sätt om det sändande användarnamnet är 'janne' och sändarens terminal är **/dev/tty03**.

```
Broadcast Message from janne (/dev/tty03) Fri Nov 24 11:01:37
                Systemet kommer att tas ned fredagen 24/11 kl 12.00
```

HÄNVISNING

write(1), mesg(1)

FELMEDDELANDEN

Ett felmeddelande visas om någon användares **tty** fil inte kan öppnas.

ANMÄRKNING

wall bör endast användas för sändning av viktiga meddelanden, såsom varning till användare om planerad systemavstängning.

NAMN

wc - räknar ord, tecken och rader.

SYNTAX

wc [-Vwcl] [filer ...]

FUNKTION

Kommandot *wc* skriver ut antalet ord, tecken och rader som finns i filerna (eller i standard input, om inga filer ges). Värden för varje fil skrivs på olika rader till standard output följt av filnamn om det har givits som argument. Om mer än en fil har angetts skrivs även det totala antalet ut på slutet. Standard är *wc* **-l** om inget tillval anges.

ord är en sträng tecken, avgränsade med mellanslag, TAB eller new-line.

TILLVAL

-V	Skriver ut versionen för kommandot *wc*.
-w	Rapporterar antal ord.
-c	Rapporterar antal tecken.
-l	Rapporterar antal rader.

EXEMPEL

```
wc myfile
5 17 86 myfile
```

för 5 rader, 17 ord och 86 tecken i filen **myfile.**

HÄNVISNING

Inga

FELMEDDELANDEN

Inga

WC(1)

NAMN

who - vem är inloggad på systemet och annan statusinformation i systemet.

SYNTAX

who [-VuTHAlpdbrtasq] [fil]

who am i

who am I

FUNKTION

Kommandot *who* listar användarens loginnamn, terminal-linje, logintid, hur lång tid som aktivitet förekommit på linjen, och kommandoavkodarens process-ID (shell) för varje användare som är aktiv i systemet. Kommandot går igenom filen **/etc/utmp** för att få denna information. Om argumentet fil anges blir den aktuella filen genomsökt. Vanligen anges då filen **/etc/wtmp**, vilken innehåller information om alla inloggningar som skett sedan filen sist skapades.

who med tillvalen **am i** eller **am I** identifierar den användare som anropar.

who används även för att lista allmän statusinformation i systemet.

Förutom standardtillvalet **-s**, är det vanligaste utskriftsformatet:

```
name (state) line time activity pid (comment) (exit)
```

Med hjälp av tillval kan kommandot *who* lista inloggning, utloggning, reboots, ändringar i systemklockan och andra processer som tagits fram av *init*-processen. Dessa tillval är:

-V Skriver ut versionen av kommandot **who**.

-u Detta tillval listar bara de användare som är inloggade för tillfället. **name** är användarens loginnamn. **line** är enheten som i biblioteket **/dev.time** är logintiden för användaren. **activity** är antalet timmar och minuter som förflutit sedan aktivitet senast förekom på den aktuella linjen. En punkt (.) anger att terminalen registrerat aktivitet den senaste minuten och därför anses aktiv. Om mer än 24 timmar förflutit eller om linjen inte använts sedan systemstart, markeras detta med texten 'old'. Detta fält kan användas för att ta reda på om någon arbetar på terminalen eller inte. **pid** är process-ID för användarens shell. **comment** är antingen det kommentarfält som angivits i filen **/etc/inittab** för aktuell terminallinje eller, om en användare är inloggad, motsvarande klartextfält från **/etc/passwd**. Informationen kan vara var terminalen finns, telefonnumret till modemet, terminaltypen, om den är direktkopplad, etc.

-T Detta tillval är detsamma som **-u** tillvalet, förutom att terminallinjens status också skrivs ut. **state** talar om

ifall det är möjligt för någon annan att skriva till terminalen. Om det är det visas ett plusstecken (+) och om det inte är möjligt visas ett minustecken (-). Root kan skriva till alla linjer som har + eller - i statusfältet. Om linjen är dålig skrivs ett frågetecken (?) ut. Om linjen är öppnad exklusivt visas tecknet **x**.

-l Detta tillval listar bara de linjer på vilka systemet inväntar login av någon. Fältet **name** är LOGIN i dessa fall. Andra fält är detsamma som med tillvalet **-u**.

-H Detta tillval skriver ut kolumnrubriker över utdataraderna.

-q Detta är en **snabbwho**, som endast visar namnen och antalet användare som är inloggade för tillfället. När detta tillvalet används blir alla övriga tillval ignorerade.

-p Detta tillval listar alla just nu aktiva processer som tidigare startats av *init*. Fältet **name** visar namnet på programmen som startats av *init* enligt filen **/etc/inittab**. Fälten **state**, **line** och **activity** har ingen betydelse. Fältet **comment** visar ID-fältet för motsvarande rad i **/etc/inittab**.

-d Detta tillval visar alla processer som avslutats och inte återstartats av *init*. Fältet **exit** skrivs ut för 'döda' processer och innehåller terminerings- och utgångsstatus returnerade från den döda processen. Tillvalet är användbart för att ta reda på orsaken till att en process terminerat.

-b Detta tillval anger tid och datum för sista systemstart (BOOT).

-r Detta tillval anger aktuell systemnivå (run-level) för processen *init*. Dessutom visas på slutet av raden de tre senaste systemnivåerna.

-A Listar information från **accounting** systemet. Används enbart tillsammans med utvecklingssystemet.

-t Detta tillval anger sista ändringen av systemklockan med hjälp av kommandot *date*, gjort av super-user (root).

-a Detta tillval listar **/etc/utmp** eller den namngivna filen, med samtliga tillval i funktion.

-s Detta tillval är standard och listar bara namnet, enheten och tidsfältet.

FILER

/etc/utmp	Nuvarande inloggade användare.
/etc/wtmp	Historiefil för inloggade användare.
/etc/inittab	

HÄNVISNING

date(1), login(1), mesg(1), su(1M), wait(1), init(1M)

FELMEDDELANDEN

Inga

ANMÄRKNING

Det är loginnamnet som visas av *who*. Det ändras inte när användaren ändrar namn till annan användare med kommandot *su*.

WHO(1)

NAMN

write - skriver direkt till annan användare.

SYNTAX

write [-V] användarnamn [tty]

FUNKTION

Kommandot *write* kopierar rader från den egna terminalen till en annan användares terminal. När *write* startas skickar det först meddelandet:

```
Message from ditt-lognamn (din-tty) [ datum ]
```

När kontakt erhållits, ges två pip-signaler (BELL) till den egna terminalen som klartecken för att börja skriva meddelandet. Mottagaren bör sedan skriva svarsmeddelande (med kommandot *write*). Kommunikationen fortgår tills end-of-file (normalt CTRL-D) läses från terminalen eller om en avbrottssignal sänds (DEL). *write* skriver då:

```
<EOT>
```

på den andra terminalen och avslutas.

Vill man skriva till en användare som har loggat in på flera terminaler kan argumentet tty användas för att ange en lämplig terminal, t.ex tty03. Om flera användare loggat in med samma namn och Du inte anger tty som argument, kommer kommandot *write* att välja första skrivbara terminal som är listad i filen **/etc/utmp** och returnera följande meddelande:

```
User is logged on more than once.
You are connected to "ttynn".
Other locations are:
ttymm
```

Tillstånd att skriva kan ges eller nekas med kommandot *mesg*. Som standard är skrivning tilllåten. Vissa kommandon, t.ex *pr*, avvisar skrivning för att förhindra dåliga utskrifter. En super-user kan använda *write* och skriva även till användare som avvisar *write*.

Om tecknet ! påträffas i början på en rad, kallar *write* in en shell för att bearbeta resten av raden som ett kommando. Detta kommando och dess utdata är av helt lokal art och skickas inte som meddelande.

Följande rutin föreslås vid användandet av *write*; när Du skrivit till en annan användare, vänta tills hon eller han skriver tillbaks innan Du börjar sända. Båda deltagarna bör avsluta respektive meddelande med en tydlig signal. (o som i "over" är konvention i engelskspråkiga länder, k som i "kom" på svenska).

Detta gör det möjligt för en annan person att svara; likaledes bör man kunna visa att konversationen är slut: oo för "over and out" på engelska, ks för "klart slut" på svenska. Det viktiga är att man är överens om signalsystemet.

TILLVAL

-V Skriver versionsnumret för kommandot *write*.

FILER

/etc/utmp för att hitta användare

HÄNVISNING

mesg(1), who(1), mail(1)

/bin/sh för att exekvera lokala kommandon med '!'.

FELMEDDELANDEN

Felmeddelanden visas om den adresserade användaren inte är inloggad eller har anropat kommandot *mesg* -n för att förhindra meddelanden (Permission denied).

En varning ges om man själv inte kan ta emot svarsmeddelanden, t ex om kommandot *mesg* -n tidigare givits.

Om mottagaren under pågående samtal ger kommandot *mesg* -n kommer *write* att upptäcka det och ge ett felmeddelande.