supermax

MS-DOS

Udvidet

Februar 1992

supermax

D E L T A G E R B E V I S

Det bekræftes herved at:

Gitte Engelsholm

Har deltaget i kurset:

MS-DOS Udvidet

I perioden:

18. til 19. marts 1992

Med venlig hilsen

Dansk Data Elektronik A/S

Jette Smith Thomsen
Kursuslærer

dde

1 Indholdsfortegnelse

1 Indledning

Dette kursusmateriale er stilet til de erfarne systemadministratorer, der gerne vil forbedre performance af sine PC'ere og uddybe sin viden om DOS i al almindelighed.

Materialet omhandler både avancerede DOS kommandoer, batchfiler, og lidt om DOS' virkemåde, og er udarbejdet på grundlag af DOS 5.0.

Mappen indeholder den teori, der gennemgås på kurset, og tillige en række øvelsesopgaver, som skal løses på kurset med instruktørens acssistance eller ved egen hjælp.

De første opgaver i hvert afsnit er for overskuelighedens skyld gjort så små som muligt. Derpå følger lidt længere opgaver.

Det er ikke tanken, at alle nødvendigvis skal nå samtlige opgaver. Meningen er, at der skal være nok til alle.

Kursusmappen kan ikke helt erstatte en DOS reference manual. I det daglige arbejde kan den dog benyttes som opslagsværk.

Kommentarer og forslag til materialet er meget velkomne på evaluerings skemaet, der udleveres på kursets sidste dag.

Herlev, den 29. oktober 1991

IO.SYS
MSDOS.SYS

Filer

MORE
SORT
FIND

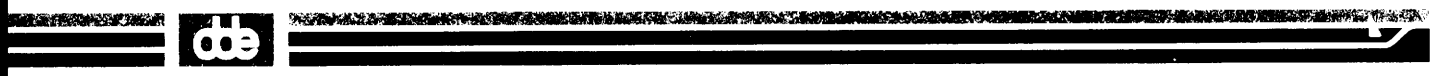
DOSKEY

2 Repetition

A:	**Første diskettedrev**
ATTRIB	**Redigere attributter for filer (Read-only, archive)**
AUTOEXEC.BAT	**Batchfil, der udføres automatisk ved boot**
BACKUP	**Sikkerhedskopiering af harddisk**
BREAK	**Bestemmer hvor ofte, der lyttes efter <Ctrl + C>**
BUFFERS	**Antallet af databuffers i RAM**
C:	**Første logiske harddisk**
CD	**Skift katalog**
CHCP	**Skift tegnsæt (codepage)**
CLS	**Slet skærm**
COM1	**Første kommunikationsport**
COMMAND.COM	**Kommandofortolker**
CON	**Konsollen (Tastatur og Skærm)**
CONFIG.SYS	**Opstartsfil, der allokerer plads i RAM til div. drivere**
COPY	**Kopier**
COUNTRY	**Sætter landekoden mht. format for dato etc.**
D:	**Anden logiske harddisk**
DATE	**Vis dato**

DEL	Slet fil
DEVICE	Indlæs driver
DIR	Vis indhold af katalog
ECHO	Skriv på skærmen
EDLIN	Rediger fil
FILES	Sætter antallet af åbne filer
FIND	Søg tegnstreng i fil
FORMAT	Formater diskette eller harddisk
IO.SYS	I/O rutiner til DOS (skjult fil)
MSDOS.SYS	Centrale dele af DOS (skjult fil)
KEYB	Sætter det nationale tegnsæt svarende til tastaturet
KEYBOARD.SYS	Driver til tastaturet
LABEL	Sæt label på drev/diskette
LPT1	Første printerudgang
MD	Opret katalog
MODE	Benyttes til styring af skærm og printer
MORE	Opdel i sider
NLSFUNC	Sprogfunktion
NUL	Skraldespand
PATH	Søgesti for kommandoer
PREPARE	Forbered tegnsæt

PRN	**Første printerudgang**
PROMPT	**Klarsignal**
RD	**Slet tomt katalog**
REN	**Omdøb fil**
RESTORE	**Re-etabler fra backup**
SORT	**Sorter linier i en fil**
TIME	**Tidspunkt**
TREE	**Vis træstrukturen**
TYPE	**Udskriv fil**
VER	**DOS version**
VOL	**Udskriv disk label**
XCOPY	**Kopier**



Opgave

Formål: - at repetere grundlæggende DOS kommandoer

1. **Boot PC'en fra systemdisketten.**

2. **Formatter øvedisketten.**

3. **Kopier filen \AUTOEXEC.BAT fra systemdisketten til øvedisketten.**

4. **Sæt read-only bitten på kopien.**

5. **Hvilke attributter gælder for kopien?**

6. **List filerne fra C:\ og gem uddata i en fil ved navn LISTE på øvedisketten.**

7. **Tilføj en listning af C:\WP51 til LISTE.**

8. **Sorter linierne i filen LISTE**

9. **Benyt FIND commandoen til at få udskrevet de underkataloger, der findes i kataloget C:\WP51**

10. **Hvilken code page er aktiv nu?**

11. **Hvilken DOS version er lagt ind på PC'en?**

12. **Opret et katalog ved navn UTIL på øvedisketten.**

13. **Flyt filen LISTE til dette katalog.**

14. **Omdøb nu filen til FILER.**

Interne kommandoer - loses ved indtastning af navn

DIR, COPY

Eksterne kommandoer

ATTRIB

kommando > fil (også LPT1 og CON)

kommando | kommando

g:p/6 bat

3 DOS kommandoer

3.1 PRINT

PRINT kommandoen bruges til at udskrive filer på printeren.

Kommandoen

PRINT AUTOEXEC.BAT CONFIG.SYS

vil udskrive filerne **AUTOEXEC.BAT** og **CONFIG.SYS** på printeren

PRINT er en såkaldt resident kommando, hvilket betyder, at den installeres i RAM lageret, første gang den kaldes.

Første gang man benytter **PRINT** efter boot, vil man få følgende spørgsmål:

Name of list device [PRN]:

Her får man mulighed for at vælge en anden printerport end LPT1, som er standard.

Efter man har valgt printerport, evt. ved at taste retur, svares:

Resident part of PRINT installed.

efterfulgt af meddelelser om køen.

Kommandoen kan betragtes som en slags baggrundsproces, så filer udskrives, mens man udfører andre kommandoer.

Når man udskriver ved hjælp af **PRINT** kommandoen, lægges de enkelte filer i kø til udskrift, da der jo højst kan udskrives én fil ad gangen på printeren.

Man kan som standard lægge op til 10 filer i kø til udskrift, men det kan sættes helt op til 32. Dette vil dog koste plads i RAM lageret.

Første gang **PRINT** kommandoen kaldes, har man mulighed for at angive en række parametre, bl.a.:

/D: **den port der skal skrives ud til**

/Q: **den maksimale størrelse print-køen kan have**

/B: **antal bytes i printbufferen i RAM lageret.**

Man kan slette et print-job fra køen ved at afgive en kommando som:

```
PRINT AUTOEXEC.BAT /C
```

der vil slette **AUTOEXEC.BAT** fra print-køen.

Hvis man vil aflæse hvilke jobs, der ligger i kø, kan man bruge kommandoen **PRINT** uden options.

Kommandoen

PRINT /T

vil slette alle jobs i køen og standse det igangværende print-job med en meddelelse:

All files canceled by operator

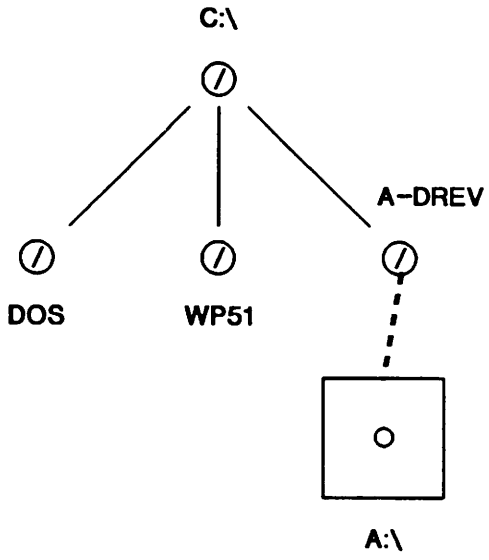
nederst på sidst udskrevne side.

3.2 JOIN

JOIN kommandoen benyttes til at "klistre" drev til kataloger.

Hvis man gerne vil kunne referere til **A:** drevet, som om det var et underkatalog på **C:**, laver man et underkatalog på **C:**, som f.eks kunne hedde **A-DREV**, og afgiver kommandoen:

JOIN A: C:\A-DREV



Så vil kataloget **C:\A-DREV** se ud som om, det indeholder hele filstrukturen fra disketten i **A:** drevet, selv om det rent fysisk stadig ligger på disketten.

En henvisning til **A:** drevet vil herefter give fejlmeddelelsen:

Invalid drive specification

Det katalog, som man "join'er" drevet til, skal være tomt, før man får lov til at lave et sådant bånd.

Man kan få listet hvilke "join's", der er gældende ved at afgive kommandoen uden argumenter.

I det foregående tilfælde vil det give en udskrift:

A: => C:\A-DREV

Et join ophæves igen ved at benytte parameteren **/D:**

JOIN A: /D

Join begrebet i DOS svarer til mount begrebet i UNIX, hvor man "klistrer" logiske subdiske på rod-disken, så det kommer til at se ud som ét sammenhængende filsystem.

Med **JOIN** kan diskstrukturen blive vanskelig at holde rede på. Man kan f.eks. ikke benytte **BACKUP** kommandoen til drev, hvor der findes join's.

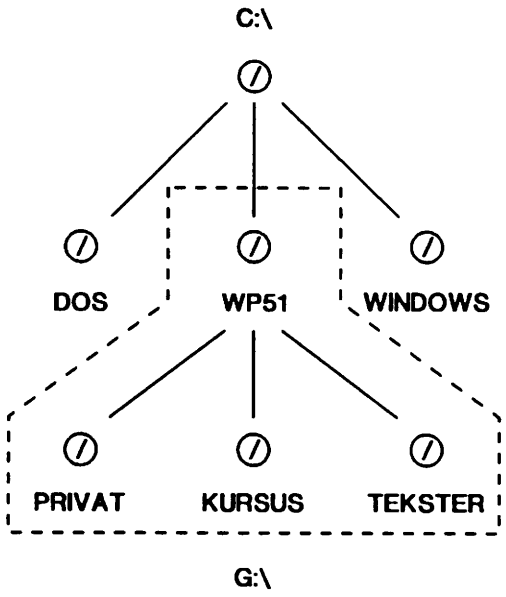
3.3 SUBST

Kommandoen **SUBST** (substitute) benyttes, hvis man vil kunne henvise til et katalog, som om det var et drev.

Efter kald af:

SUBST G: C:\WP51

vil **G:** drevet se ud, som om det indeholder det, der ligger i **C:\WP51** kataloget:



Hvis man afgiver kommandoen **SUBST** uden argumenter, vil man få listet de substitutioner, der er gældende.

Eksemplet fra før vil give følgende udskrift:

SUBST

G: => C:\WP51

Man kan ophæve en gældende substitution ved at benytte option /D til kommandoen:

SUBST G: /D

Man kan ikke substituere det aktuelle drev med en sti, og man skal være forsigtig ved backup, hvis man har substituerede drev.

Opgave

Formål: - at afprøve kommandoerne JOIN, SUBST og PRINT

1. **Opret et nyt katalog ved navn C-DREV på øvedisketten, og list indholdet af dette katalog.**
2. **JOIN C: drevet til dette drev, og se med JOIN uden parametre, at det er gået godt.**
   Hvorfor giver det nu problemer?
   Hvad skal der til for at få det til at gå godt?
3. **List nu indholdet af kataloget igen.**
4. **Opret en fil ved navn A:\C-DREV\NYFIL.**
5. **Slet linket til C: drevet, og list nu indholdet af kataloget C-DREV.**
   Hvad blev der af filen NYFIL? Find den og slet den igen.
6. **Giv kataloget A:\DOS på systemdisketten navnet G:**
7. **Skift ned til kataloget A:\DOS. Hvad sker?**
8. **Skift nu til G: drevet, og list indholdet her.**
9. **List de gældende SUBST's**
10. **Boot og list igen de gældende SUBST.**

Ekstraopgave

Formål: - at **afprøve PRINT kommandoen**

1. **Gå til en af de PC'ere, som instruktøren anviser.**

2. **Benyt kommandoen**

   MEM /C

   til at undersøge hvad der ligger i RAM lageret.

   Kig specielt efter "PRINT".

3. **Udskriv filen C:\AUTOEXEC.BAT ved hjælp af PRINT kommandoen.**

   Kig igen efter "PRINT" i RAM.

4. **Skriv en stor fil ud, og prøv at se, om du kan nå at se indholdet af printkøen.**

5. **Udskriv nu flere filer på en gang og list køen igen.**

4 Batchfiler

En batchfil er simpelthen en samling af DOS kommandoer, der er gemt i en fil.

En simpel batchfil kunne være:

```
CLS
VER
VOL
```

Denne batchfil vil slette skærmen, udskrive på skærmen, hvilken DOS udgave der benyttes, og labelen fra det aktuelle drev.

Hver kommandolinie i filen vil blive udskrevet på skærmen i takt med, at kommandoerne udføres.

Generelt gælder følgende regler for navngivning af batchfiler:

* **Brug ikke navne på interne DOS kommandoer, de interne kommandoer vil blive udført i stedet for batchfilen**

* **Batchfiler skal altid have filtypen BAT og kan udføres blot ved at angive filnavnet**

* **DOS vil udføre filer af typen COM eller EXE, hvis der findes filer med denne type og med samme navn som batchfilen**

Man kan kalde én batchfil fra en anden batchfil, men hvis det ikke sker i sidste linie batchfilen, skal man kalde kommandofortolkeren (**COMMAND.COM**), før man kalder næste batchfil som i dette eksempel:

```
CLS
DATE
COMMAND /C BATCH
CLS
```

Her udføres en batchfil ved navn **BATCH.BAT** af en sekundær kommandofortolker.

Hvis ikke denne fremgangsmåde benyttes, vil kommandofortolkeren ikke vende tilbage til den første batchfil og fortsætte udførelsen.

4.1 Bemærkninger

Batchfiler kan meget hurtigt blive meget uigennemskuelige, selv for den, der har skrevet dem. Derfor er det vigtigt, at man skriver forklarende bemærkninger ind i dem.

DOS kommandoen **REM** giver adgang til at skrive en meddelelse på op til 123 tegn på resten af linen.

REM kommandoer bliver også vist på skærmen, når batchfilen udføres, og således kan man få meddelelser skrevet ud til brugeren. Dette gælder dog ikke, hvis man har slået **ECHO** fra.

4.2 ECHO kommandoer

En pænere måde at udskrive kommentarer til brugerne er ved at bruge **ECHO** kommandoen.

ECHO kommandoen har flere funktioner. Man kan benytte **ECHO** til at slå udskrivning af kommandolinierne fra og til ved at benytte hhv.

```
ECHO OFF

ECHO ON
```

Selve linien med **ECHO OFF** udskrives på skærmen, men resten af linierne vil ikke blive udskrevet, før der evt står en **ECHO ON** kommando.

Hvis man ikke ønsker, at **ECHO OFF** kommandoen udskrives, kan man skrive et "@" først på linen:

```
@ECHO OFF
```

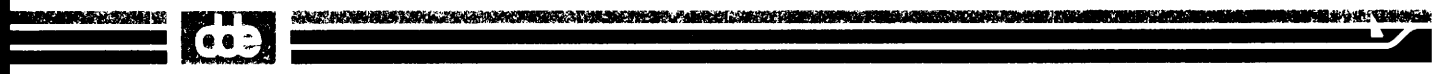
så udskrives linien ikke på skærmen under udførelsen.

Denne fremgangsmåde kan benyttes til at forhindre en hvilken som helst linie i at blive udskrevet på skærmen, hvis man ikke benytter **ECHO OFF** og **ECHO ON**.

Når man har slået ekko fra, kan man udskrive kommentarer på maksimalt 117 tegn til brugeren ved benytte **ECHO** kommandoen efterfulgt af kommentaren:

```
ECHO Nu udføres næste kommando.
```

Kun selve kommentaren vil så blive vist på skærmen under udførelsen.



4.3 PAUSE

Kommandoen **PAUSE** kan også benyttes til at udskrive meddelelser til brugeren.

```
PAUSE Nu bliver kataloget slettet
```

vil give en udskrift på skærmen:

```
C:\>PAUSE Nu bliver kataloget slettet
Strike a key when ready . . .
```

Udførelsen af batchfilen fortsætter først, når brugeren taster retur, ellers kan brugeren afbryde udførelsen ved at taste ^C.

Hvis man har slået **ECHO** fra, vil meddelelsen dog ikke blive skrevet ud.

4.4 Oprettelse af batchfiler

Hvis man vil oprette en batchfil, kan man benytte en editor. Standard editoren i DOS hed tidligere **EDLIN**, og UNIX editoren **vi** findes
også i en DOS udgave, der dog ikke i alle udgaver kan håndtere de danske bogstaver **Æ**, **Ø** og **Å**.

Fra og med DOS 5.0 findes en ny skærmorienteret editor, **EDITOR**, der er standard i DOS, men den "gamle" editor **EDLIN** kan stadig godt bruges.

En lidt mere primitiv måde at danne filer på er at benytte kommandoen:

COPY CON: FIL

der kopierer fra **CON:** (tastaturet) til en fil ved navn **FIL.** Hvis filen **FIL** allerede eksisterer, vil den blive overskrevet.

Man kan benytte en lignende konstruktion til at tilføje til allerede eksisterende filer:

COPY FIL + CON:

Hvis man ønsker at redigere i allerede eksisterende filer, skal man benytte en editor til dette.

Man kan ikke benytte et tekstbehandlings program til at foretage redigering af batchfiler, da batchfiler skal lagres i det, der hedder ASCII format, og tekstbehandlingsprogrammer lagrer som regel i et andet format, der bl.a. indeholder tabulator linealer, valg af skrifttype og printer etc.

I de fleste tekstbehandlingsprogrammer har man dog mulighed for at lagre tekster i ASCII format.

I WordPerfect gøres det f.eks. ved at vælge **Tekst ind/ud** (<Ctrl + f5>) efterfulgt af **1.** for **DOS tekst** og **1** for **Gem.**

Opgave

Formål: - at skrive og udføre simple batchfiler
- at afprøve kommandoerne ECHO, REM og PAUSE

1. **Tast ECHO OFF fra DOS prompten.**
   Hvad sker?
   Ret det op igen.

2. **Lav en lille batchfil, ECHOTEST.BAT, der udskriver dagens dato med en ECHO sætning (ikke ved brug af DATE), og derefter holder en pause.**

   PAUSE kommandoen skal kaldes med en kommentar: "Tast retur, hvis du vil videre".

   Kør filen, og se resultatet.

   Hvordan kunne man smide det ordinære uddata fra PAUSE kommandoen væk?

   Slå echo til skærmen fra i første linie i filen, og kør den én gang til.

   Hvordan gik det med PAUSE kommandoen.

3. **Skriv en ny batchfil, der starter med at slette skærmen, derefter kalder ECHOTEST og til slut sletter skærmen igen.**

   Kør filen.

   Fik du slettet skærmen til slut?

5 Parametre

Hvis man ønsker at overføre værdier til en batchfil, kan man benytte parametre.

En parameter er en værdi, der angives efter navnet på batchfilen, på kommandolinien.

TYPE PARAM.BAT

```
@echo off
echo Dette er første parameter %1
echo Dette er anden parameter %2
echo Dette er tredie parameter %3
```

PARAM FØRSTE ANDEN TREDIE

```
Dette er første parameter: FØRSTE
Dette er anden parameter: ANDEN
Dette er tredie parameter: TREDIE
```

Som det ses af eksemplet, bliver **%1** sat til det første ord efter navnet på batchfilen, **%2** til det næste og så fremdeles til **%9**, som altså får værdien af det niende ord efter kommandoens navn i kaldet.

Hvis man ønsker at benytte mere end 9 argumenter, må man starte batchfilen med at gemme værdierne af **%1** til **%9**, og så kan man benytte kommandoen **SHIFT**, der skifter parameterne en plads til venstre.

Eksempel:

```
PARAM A B C D E F G H I J K L M O P Q
```

Før brug af **SHIFT**:

%1	%2	%3	%4	%5	%6	%7	%8	%9
A	B	C	D	E	F	G	H	I

Efter brug af **SHIFT**:

%1	%2	%3	%4	%5	%6	%7	%8	%9
B	C	D	E	F	G	H	I	J

Hvis man har flere end 9 parametre, vil man altså kunne få fat i det tiende ved at henvise til **%9** efter at have brugt **SHIFT** kommandoen én gang.

%0 er en særlig parameter, der indeholder navnet på batchfilen. Hvis man benytter **SHIFT** kommandoen, vil **%0** komme til at indeholde værdien af første parameter.

5.1 SET kommandoen

Hvis **SET** kaldes uden brug af argumenter, vil man få udskrevet alle variable fra parameterblokken:

```
COMSPEC=C:\COMMAND.COM
PATH C:\;C:\DOS;C:\UTILS
PROMPT=$P$G
```

Man kan benytte **SET** kommandoen til at definere nye variable:

```
SET TEMP=C:\WINDOWS\TEMP
```

vil definere en ny varibel, **TEMP** som er sat til **C:\WINDOWS\TEMP.**

Hvis man skal benytte værdien af **TEMP** i en batchfil, kan det gøres på følgende måde:

```
COPY FIL.TMP %TEMP%
```

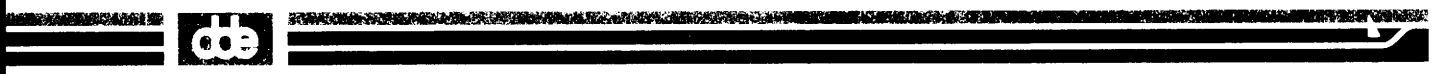
Hvis man skal benytte en variabel meget, kan man evt sætte den i **AUTOEXEC.BAT.**

Herefter kan man benytte den fra alle batchfiler.

Hvis man definerer variable i batchfiler, så bliver de stående i parameterblokken, også efter batchfilen er afviklet, lige indtil kommandofortolkeren forlades med **EXIT** eller PC'en bootes.

Man kan dog også slette en variabel fra parameterblokken ved at give en kommando:

```
SET TEMP=
```



Opgave

Formål: - at oprette og afprøve batchfiler med parametre

1. **Afprøv eksemplerne fra teksten.**

2. **Tilføj en SHIFT kommando til PARM.BAT og lad filen udskrive parametrene igen.**

3. **Skriv en batchfil, der tager to argumenter: en fil og et katalog. Batchfilen skal flytte filen fra det aktuelle katalog til det angivne katalog.**

   Dette skal gøres ved først at kopiere filen til kataloget, og derefter skal filen slettes fra det aktuelle katalog.

4. **Skriv filen om, så brugeren spørges om den oprindelige fil skal slettes, inden dette gøres. (Brug PAUSE kommandoen).**

6 Makroer

En makro er en samling DOS kommandoer, der kan udføres ved at taste navnet på makroen. I modsætning til en batchfil, gemmes en makro i RAM lageret og ikke i en fil. Det vil sige, at når PC'en bootes forsvinder makroerne.

En makro oprettes f.eks. således:

```
DOSKEY KOPI=COPY *.* A:
```

Denne makro får navnet **KOPI** og kopierer alt fra arbejdskataloget til disketten i drev a:.

Hvis man ønsker at oprette en makro, der består af flere kommandoer, kan man gøre det ved at indsætte et **$** (dollartegn) efterfulgt af et **"T"** mellem kommandoerne:

```
DOSKEY KOPI2=DIR  $T COPY *.* A:
```

Antallet af kommandoer i en makro er der ingen begrænsning på, men hele makroen må højst være på 127 tegn.

$T ↵ eff. ...
$G >
$L <
$B |

6.1 Brug af parametre i makroer

Man kan benytte parametre i makroer ligesom i batchfiler. I en makro benyttes i midlertid betegnelserne **$1**, **$2**, ..., **$9** for parametrene.

```
DOSKEY FLYT=COPY $1 $2 $T DEL $1
```

Denne makro vil kopiere første parameter (**$1**) til det sted, der er angivet i anden parameter (**$2**), og derefter slette første parameter.

Hvis man ønsker at henvise til resten af linien efter selve makronavnet i kommandolinien, kan man benytte parameteren **$***.

Følgende makro vil således udskrive alt, hvad der står efter **SKRIV** i kommandolinien, og fungerer således som DOS kommandoen **ECHO**:

```
DOSKEY SKRIV=echo $*
```

6.2 Listning af makroer

Hvis man ønsker at få listet alle de makroer, der er lagret i RAM, kan man skrive:

```
DOSKEY /MAKROS
```

Som vil give en udskrift:

```
KOPI=COPY *.* A:
KOPI2=DIR $T COPY *.* A:
FLYT=COPY $1 $2 $T DEL $1
SKRIV=ECHO $*
```

6.3 Sletning af Makroer

Hvis man vil slette makroen **SKRIV**, skriver man:

DOSKEY SKRIV=

Og man kan slette samtlige makroer ved at skrive:

<Alt>+<f10>

Som tidligere beskrevet slettes samtlige makroer ved boot. Har man lavet et antal makroer, som man ønsker skal bruges, hver gang man booter, kan man lægge definitionerne af dem i **AUTOEXEC.BAT** eller læse dem over i en batchfil på følgende måde:

DOSKEY /MACROS > MAKROER.BAT

I denne batchfil kan man tilføje **DOSKEY** foran hver af linierne. Denne batchfil kan så benyttes, når man vil definere makroerne.

6.4 Omdirigering af input og output

Hvis man ønsker at benytte omdirigering i en makro, kan man ikke benytte de sædvanlige tegn ">" eller "<", men man må i stedet benytte hhv. **$G** og **$L**.

Følgende makro lister filerne i det aktuelle katalog og gemmer uddata i filen **LISTE**:

```
DOSKEY LIST=DIR $G LISTE
```

og følgende lister indholdet af en den fil, der angivet som første parameter, opdelt i sider vha **MORE** kommandoen:

```
DOSKEY PG=MORE $L $1
```

6.5 Brug af pipes i makroer

Man kan godt benytte pipes i en makro, men hvis man benytter det sædvanlige pipetegn, "¦", vil det blive betragtet som om, at det er uddata fra **DOSKEY** kommandoen, der omdirigeres. Derfor benyttes tegnet "**$B**":

```
DOSKEY  SORT $L $1 $B MORE
```

Her sorteres linierne i filen angivet i første parameter og uddata udskrives sidevis ved at pipe over til **MORE** kommandoen.

Opgave

Formål: - at udfærdige og afprøve makroer

1. **Afprøv de makroer, der er beskrevet i teksten.**

2. **Lav makroen KOPI om, så der først udskrives en besked til brugeren, før filerne kopieres.**

3. **Skriv en makro, HVOR, der søger et angivet drev for at finde en angiven fil. Du kan evt. benytte DIR kommandoen med option /s til søgningen.**

   Kaldet af makroen kunne f.eks. se således ud:

   ```
   HVOR C: WIN.INI
   ```

   hvis man ønsker at finde filer med navnet WIN.INI på C: drevet.

4. **Skriv en makro, der udskriver linierne i en fil sorteret og gemmer uddata i en fil.**

5. **Skriv en makro, der finder de linier i filen LISTE fra eksemplet, der indeholder ordet EXE sorterer dem og skriver dem en side af gangen.**

7 Kontrolstrukturer

I DOS findes, som i de fleste programmeringssprog, mulighed for at udføre betingede sætninger. Det vil sige sætninger, hvis udførelse afhænger af en given betingelse.

7.1 IF sætninger

I DOS ser syntaxen for en **IF**-sætning således ud:

IF *betingelse kommando*

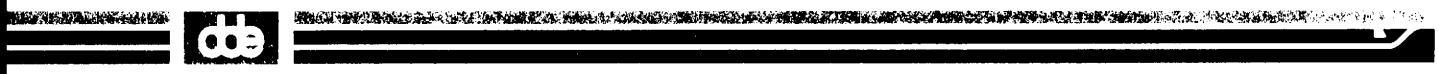
Eks.

IF NOT EXIST AUTOEXEC.BAT ECHO AUTOEXEC.BAT eksisterer ikke

Her testes om **AUTOEXEC.BAT** eksisterer. Hvis ikke vil teksten

AUTOEXEC.BAT eksisterer ikke

blive skrevet til skærmen.



Betingelser i DOS kan se ud på følgende måde:

EXIST *fil*
NOT EXIST *fil*
streng == *streng*
NOT *streng* == *streng*
ERRORLEVEL #
NOT ERRORLEVEL #

Streng er en hvilken som helst samling af tegn, dog skal begge strenge være omgivet af ", hvis en af dem indeholder mellemrum.

ERRORLEVEL tester på exitkoden fra sidst udførte DOS kommando. Visse DOS kommandoer returnerer en exitkode på 0 eller derover. Generelt betyder exitkode 0, at kommandoen gik godt, og exitkoder på 1 eller derover betyder, at kommandoen fejlede.

Exitkoder for DOS kommandoer:

BACKUP	**0**	**Normal afslutning**
	1	**Ingen filer fundet til kopiering**
	2	**Visse filer ikke kopieret pga. deling i netværk**
	3	**Afbrudt af brugeren**
	4	**Afbrudt pga. fejl.**
FORMAT	**0**	**Normal afslutning**
	3	**Afbrudt af brugeren**
	4	**Afbrudt pga. fejl**
	5	**Afbrudt pga. svar "nej" til formatering af fast disk**

REPLACE	**2**	**Filen blev ikke fundet**
	3	**Kataloget blev ikke fundet**
	5	**Filen er Read-only og /R er ikke angivet**
	8	**For lille lager til kommandoen**
	11	**Kommandoens parametre er skrevet forkert**
RESTORE	**0**	**Normal afslutning**
	1	**Ingen filer fundet af genlagre**
	2	**Visse filer ikke genlagret pga. deling i netværk**
	3	**Afbrudt af brugeren**
	4	**Afbrudt pga. fejl**
DISKCOPY	**0**	**Normal afslutning**
	1	**Uoprettelig fejl**
	2	**Afbrudt af brugeren**
	3	**Fatal disk fejl**
	4	**Syntaxfejl**

Hvis kommandoen returner en exitkode på mere end eller lig med #, vil betingelsen "**ERRORLEVEL**" være sand.

Eksempel 1:

```
IF "%1" == "" echo Der skal angives en parameter!!
```

Her testes, om der er angivet en parameter til scriptet. Hvis der ikke er (**%1** er den tomme streng), udskrives sætningen.

Eksempel 2:

```
IF NOT ERRORLEVEL 0 ECHO Kommandoen gik ikke godt.
```

7.2 GOTO

GOTO sætninger sammen med labels benyttes til at styre program-forløbet i en batchfil.

Syntaxen ser således ud:

GOTO *label*

hvor en label er en måde at identificere en bestemt linie i filen.

En label angives ved at skrive et kolon efterfulgt af et ord på en linie.

Alt, hvad der kommer efter labelen på linien, vil blive ignoreret.

Et sted, hvor man typisk benytter **GOTO** sætninger, er i forbindelse med **IF**-sætninger.

På denne måde kan man få udført en hel blok af kommandoer, hvis betingelsen er sand, og en anden blok af sætninger, hvis betingelsen er falsk.

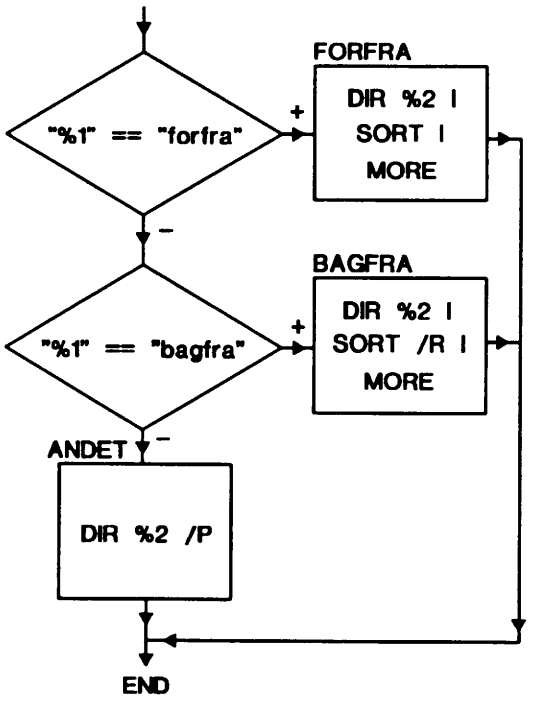
Eksempel 3:

```
IF "%1" == "forfra" goto FORFRA
IF "%1" == "bagfra" goto BAGFRA
GOTO ANDET

:FORFRA
     DIR %2 ¦ SORT ¦ MORE
     GOTO END

:BAGFRA
     DIR %2 ¦ SORT /R ¦ MORE
     GOTO END

:ANDET
     DIR %2 /P
     GOTO END

:END
```

Denne batchfil udskriver en listning af det katalog, der er angivet i anden parameter.

Hvis første parameter er **forfra** sorteres filerne almindeligt, og hvis det er **bagfra**, sorteres filerne i omvendt rækkefølge. I alle andre tilfælde udskrives blot en usorteret kataloglistning.

Forløbet i denne batchfil kan illustreres med følgende rutediagram:



Hvis ikke **GOTO** benyttes på en struktureret måde, vil batchfilerne meget hurtigt blive meget uoverskuelige.

Så en hovedregel er, at man aldrig skal benytte flere **GOTO** kommandoer end absolut nødvendigt.

Opgave

Formål: - at afprøve kontrolstrukturer

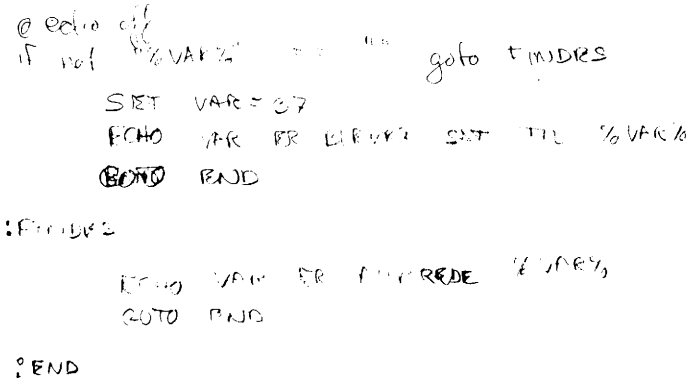
1. **Afprøv eksemplerne i teksten.**

2. **Skriv følgende pseudo-kode om til en batchfil:**

   Fjern udskrift til skærm
   Hvis variablen VAR er defineret, og ikke-tom:
   udskriv værdien af den,
   ellers
   sæt den til 34
   og udskriv en meddelelse om at det er gjort.

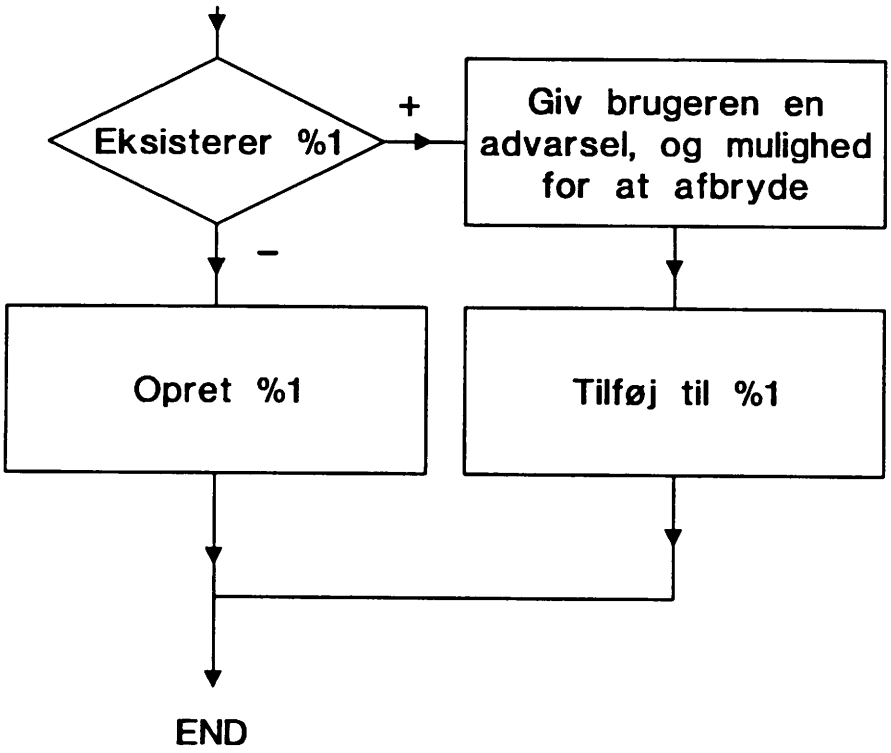
3. **Afprøv batchfilen både hvor VAR har værdien 27, og hvor VAR ikke har fået nogen værdi.**

4. **Skriv et rutediagram til batchfilen.**



.. *opgaven fortsættes*

5. Lav en batchfil, der svarer til dette rutediagram:



Dette rutediagram viser en batchfil til at oprette tekstfiler. Hvis der allerede eksiterer en fil med det angivne navn, skal der tilføjes til den eksisterende fil.

Advarsel til brugeren gives ved hjælp af en ECHO sætning, og mulighed for at afbryde kan man give ved at benytte kommandoen PAUSE.

Husk at man tilføjer til en allerede eksisterende fil, ved at benytte kommandoen

```
COPY %1 + CON:
```

og at man opretter en ny fil ved kommandoen

```
COPY CON: %1
```

8 Løkker

I DOS findes også mulighed for at få gentaget en række kommandoer.

Dette kan gøres ved hjælp af en **FOR** løkke, hvor man får afviklet en kommando et givent antal gange, eller ved hjælp af betingede løkker, hvor antallet af gange løkken afvikles, afhænger af en betingelse.

8.1 FOR løkker

Syntaksen for en FOR løkke ser således ud:

FOR %%*variabel* **IN** (*værdier*) **DO** *kommando*

Denne fungerer på følgende måde:

Først sættes variablen lig med den første af værdierne, og kommandoen udføres, derefter sættes variablen lig med den næste værdi, og kommandoen udføres igen, og så fremdeles.

Et variabelnavn kan være et hvilket som helst bogstav mellem A og Z.

Eksempel 1:

```
FOR %%F IN (AUTOEXEC.BAT CONFIG.SYS  TEST.BAT) DO TYPE %%F
```

Her vil hver af filerne **AUTOEXEC.BAT**, **CONFIG.SYS** og **TEST.BAT** blive udskrevet på skærmen.

Hvis man vil lave den samme løkke interaktivt, dvs. fra kommandolinien i stedet for fra en batchfil, ser syntaksen lidt anderledes ud:

```
FOR %F IN (AUTOEXEC.BAT CONFIG.SYS TEST.BAT) DO TYPE %F
```

altså med kun ét "%" foran variabelnavnet.

8.2 Betingede løkker

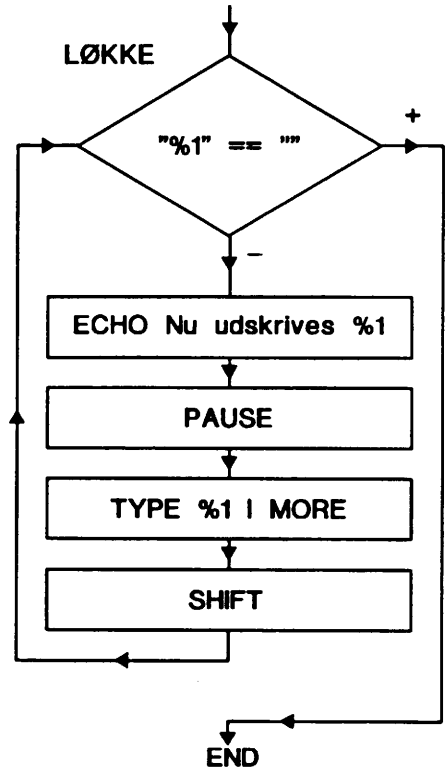
Man kan også kombinere brugen af **GOTO** med brugen af **IF**-sætninger, og lave løkker på denne måde:

Eksempel 2:

```
ECHO OFF
:LØKKE
IF "%1" == "" GOTO END
        ECHO Nu udskrives %1
        Pause
        TYPE %1 ¦ MORE
        SHIFT
        GOTO LØKKE

:END
```

Bemærk hvordan blokken af kommandoer fra lablen **LØKKE** er rykket en tand til højre for at vise strukturen i programmet.

Denne løkke kan illustreres ved følgende rutediagram:



Opgave

Formål: - *at skrive FOR løkker*
- *at lave løkker ved hjælp af GOTO og IF*

1. **Afprøv de løkker, der er beskrevet i teksten.**

2. **Ret FOR løkken fra Eksempel 1 til, så filerne ikke udskrives, men der søges efter de linier, der indeholder tallet 865.**

   for %%f in (autoexec.bat config.sys) do find "865" %%f

3. **Lav løkken i Eksempel 2 om, så de angivne filer ikke udskrives, men kopieres til et katalog ved navn UTIL på øvedisketten.**

   copy %f A:\UTIL\%f

   Husk at oprette kataloget, inden du kører filen.

   Hvordan fungerer batchfilen med wildcards? fungerer ikke

4. **Skriv en batchfil, der ved hjælp af en FOR løkke udskriver tallene fra 0 til 10.**

   FOR %%E IN (0 1 2 3 4 5 6 7 8 9 10) DO ECHO %%E

9 Repræsentation af tal

Repræsentation af tal kan foregå på mange forskellige måder. De fleste lande benytter efterhånden alle 10-talsystemet, hvor man har cifrene 0, 1, 2, 3, 4, 5, 6, 7, 8 og 9.

For at kunne forstå andre talsystemer er det vigtigt, at man forstår virkemåden af 10-talsystemet helt til bunds.

Lad os studere tallet 5896. Hvis man begynder fra højre, så får man:

6 1'ere	=	6
9 10'ere	=	90
8 100'ere	=	800
5 1000'ere	=	5000
Sum	=	5896

Summen af disse tal giver altså netop 5896.

Hvorfor ganger man så med 1, 10, 100 og 1000?

Det er fordi, vi taler om 10-tals systemet, og så skal cifret på pladsen længst til højre ganges med 10^0, som er 1. Næste ciffer skal ganges med 10^1, altså med 10, og næste igen med 10^2, som er 100, og så fremdeles.

Et ciffers position bestemmer altså hvilken værdi, det står for.

9.1 Binære tal

En hver datamat vil, når man går helt ned på laveste niveau, udføre alle operationer ved hjælp af transistorer, der har strøm eller ikke har strøm. Dette bruges til at repræsentere hhv. 1-taller og nuller, og man kalder det så for bit.

Når man samler flere sådanne bit, kan man repræsentere tal ved at benytte det binære talsystem, hvor de eneste cifre, der findes er 1-taller og nuller.

F.eks. vil det binære tal 101010 svare til det decimale tal:

0 1-ere	(2^0)=	0
1 2-er	(2^1)=	2
0 4-ere	(2^2)=	0
1 8-er	(2^3)=	8
0 16-ere	(2^4)=	0
1 32-er	(2^5)=	32
Ialt		42

Her starter man igen fra højre med at gange med hhv. 2^0, 2^1, 2^2, 2^3, 2^4 og 2^5.

Tager man 4 bit (en halv byte) vil man altså have mulighed for at repræsentere tallene fra 0 til 1+2+4+8=15, og så har man mulighed for at benytte 16-talsystemet, eller det man kalder hexadecimale tal.

9.2 Hexadecimale tal

Normalt benytter vi 10-talsystemet, hvor man har cifrene 0-9, så hvis man skal skrive hexadecimale tal, kommer man til at mangle repræsentation af "cifrene" 10, 11, 12, 13, 14 og 15.

Dette klarer man ved at benytte bogstaverne A, B, C, D, E og F, så

A betyder 10
B betyder 11
C betyder 12
D betyder 13
E betyder 14
F betyder 15

Hvordan får man så "oversat" et hexadecimalt tal til decimal-tal?

Hvis vi ser på det hexadecimale tal F7AE, så betyder det:

E gange 16^0,	dvs. 14 gange	1=	14
A gange 16^1,	dvs. 10 gange	16=	160
7 gange 16^2,	dvs. 7 gange	256=	1792
F gange 16^3,	dvs. 15 gange	4096=	61440
Ialt			63406

Til slut kan nævnes, at hvis man tager 3 bit sammen under ét, får man mulighed for at repræsentere tallene fra 0 til 1+2+4=7, og så kan man repræsentere oktale tal (otte-talsystemet).

Opgave

Formål: - at blive lidt mere fortrolig med hexadecimale tal.

Tabel: $16^0 = 1$
$16^1 = 16$
$16^2 = 256$
$16^3 = 4096$

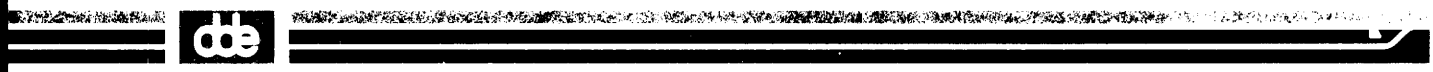
1. **Hvad kræver flest cifre: et hexadecimalt tal eller det tilsvarende decimale eller binære?**

2. **Omsæt det decimale tal 23 til hexadecimalt.**

3. **Hvad er hexadecimalt ABC decimalt? (Du behøver ikke udregne tallet, bare angive en formel.**

4. **Hvor store tal kan man angive med 3 cifrede hexadecimale tal. (Du behøver ikke udregne tallet).**

5. **Hvilke cifre skal man bruge for at repræsentere oktale tal (otte-talssystemet)?**



10 Organisation af diske

10.1 Formatering af disketter

Når en diskette formateres af DOS, bliver den delt op i et antal sektorer, hver på 512 bytes.

I den første sektor lagres bootprogrammet. Dette indeholder en række oplysninger om diskettens kapacitet.

De næste sektorer benyttes til en tabel over diskettens indhold. Denne tabel kaldes **File Allocation Table** eller i daglig tale: **FAT** tabellen.

Den indeholder oplysninger om diskettens indhold af filer og kataloger.

Denne tabel er meget vigtig, da man slet ikke vil kunne bruge disketten uden den, så derfor er der faktisk to kopier af tabellen.

Størrelsen af FAT tabellen varierer fra diskettetype til diskettetype.

De følgende sektorer på disketten indeholder **rod kataloget** med oplysninger om filnavne, hvor på disketten filerne starter, filernes størrelse, tidspunkt for sidste ændring og filernes egenskaber.

Oplysningerne om en enkelt fil fylder 32 bytes, så hver sektor i dette katalog kan altså indeholde oplysninger om 16 filer.

Alle **underkataloger** betragtes af DOS som filer, der indeholder oplysninger om andre filer, og som er mærkede på en speciel måde.

Hvor mange sektorer, der er afsat til rod kataloget, afhænger igen af diskettetypen, men det er klart, at antallet af filer og underkataloger i rod kataloget vil afhænge af denne plads.

For 5 ¼ tomme disketter gæder følgende tal:

Single Sided, Double Density: max 64 filer/kataloger i roden
Double Sided, Double Density: max 112 filer/kataloger i roden
Double Sided, High Density: max 224 filer/kataloger i roden

Og for 3½ tomme disketter gælder:

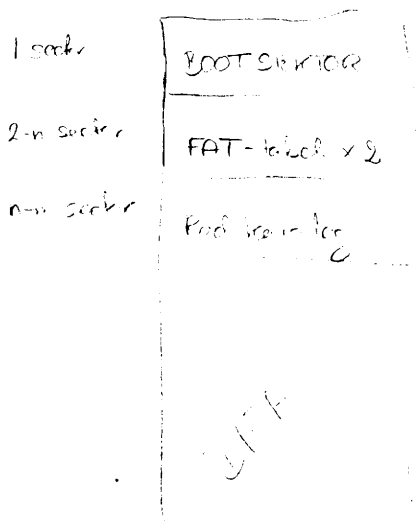
Double Sided, Double Density: max 112 filer/kataloger i roden
Double Sided, Quad Density: max 224 filer/kataloger i roden

Da underkataloger blot er almindelige filer, der er mærkede, gælder der ingen særlige begrænsninger for antallet af filer i et underkatalog.

Resten af sektorerne på disketten benyttes til data område, hvor almindelige filer og underkataloger lagres.

På de fleste diskettetyper grupperes sektorerne to og to i såkaldte clustre eller klynger, således at en fil altid fylder mindst én klynge, eller 1024 bytes (1Kb). Dog vil et klynge på en 3½" diskette på 1,44 Mb være af størrelsen 512 bytes eller kun én sektor.

Alle sektorer til filer overskrives ved formatering af disketten med et specielt tegn med ASCII koden 246 (÷). Derfor kan man ikke på nogen måde genfinde filer fra en diskette efter formatering.



10.2 Formatering af faste diske

En fast disk leveres altid formateret, og i den første sektor findes et specielt Master Boot Program.

Brugerens første opgave er at dele disken ind i logiske disk-drev ved hjælp af kommandoen **FDISK**. Oplysningerne om denne opdeling bliver skrevet i en tabel lige efter Master Boot Programmet.

Herefter skal brugeren formatere hver af de logiske diske ved hjælp af **FORMAT** kommandoen.

I modsætning til disketter skrives der kun i den første byte af klyngerne, så filer kan rekonstrueres ved hjælp af diverse hjælpeprogrammer efter formatering.

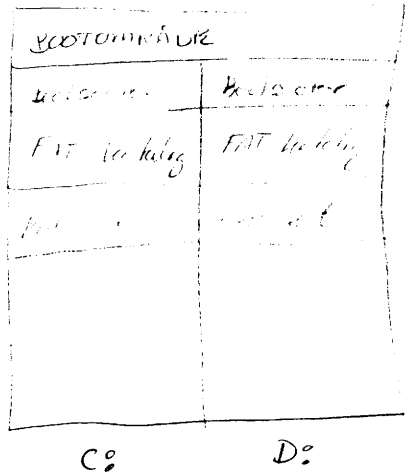
Ved formatering gøres plads til en FAT tabel, rod kataloget og bootsektoren ligesom ved formattering af disketter.

På en fast disk vil en klynge typisk bestå af 4 sektorer, og dermed vil en fil fylde mindst 4 gange 512 bytes eller 2 Kb.

DOS adresserer sektorerne på disken ved hjælp af 4-cifrede hexadecimale tal (2 bytes), så DOS 3.30 kan altså ikke adressere flere end 16^4=65536 sektorer, så den maksimale diskstørrelse i DOS 3.3 bliver altså på 65536 gange 512 bytes pr. sektor, ialt 32 Mb.

Har man en fysisk disk på mere end 32 Mb, skal denne altså deles op i logiske drev hver på maksimalt 32 Mb.

Fra DOS 5.0 kan man dog benytte logiske drev på op til 2 Giga bytes.



10.3 Rod kataloget

I rod kataloget findes, som tidligere nævnt, en beskrivelse af hver fil på disken. Herunder placeringen af filnavnet, filattributterne, dato for sidste ændring af filen og nummeret på den klynge, hvor den første sektor af filen ligger gemt, samt en pegepind til FAT-tabellen.

Der er 32 bytes til at beskrive den enkelte fil. Filnavnet fylder 11 bytes (8 til selve navnet og 3 til filtypen). Den næste byte benyttes til at beskrive filens attributter.

En byte består som bekendt af 8 bit, og disse er udnyttet på følgende måde:

00000001	Read-only bitten
00000010	Filen er skjult/ikke-skjult
00000100	Filen er en systemfil/ikke en systemfil
00001000	Filen er en disk-etiket
00010000	Filen er et katalog
00100000	Arkiv bitten
01000000	Ikke brugt
10000000	Ikke brugt

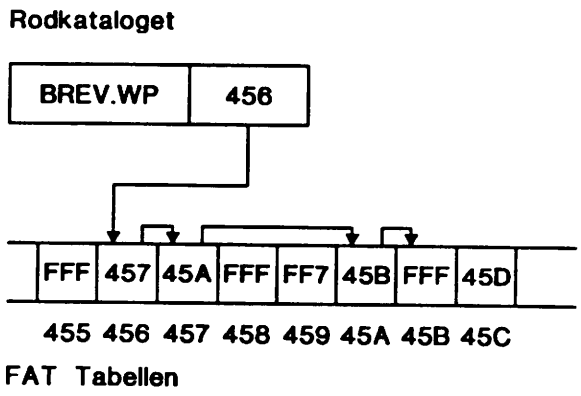
Således vil 00100111 (27 hexadecimalt) betyde, at filen er en skrivebeskyttet, skjult, systemfil, der skal tages med ved næste sikkerhedskopiering.

10.4 FAT Tabellen

Når filer lagres på disken, vil de ikke altid kunne ligge lagret i på hinanden følgende klynger.

For at kunne finde de klynger, filen består af, benyttes FAT tabellen, der består af 3-cifrede hexadecimale tal; ét for hver klynge på disken.

Eksempel:



Her ses, at filen **BREV.WP** starter i klyngen med nummer 456, og fortsætter i 457, 45A til 45B.

Enhver fil modsvares således af en "kæde" i FAT tabellen.

En kæde slutter altid med det hexadecimale tal FFF, der markerer, at den tilhørende klynge er ledigt.

Tallet FF7 i et felt angiver, at de tilhørende klynger er defekte.

10.5 CHKDSK

Kommandoen benyttes uden parametre til at kontrollere et drev med hensyn til diskens label, tidspunkt for sidste ændring af lablen, diskens totale størrelse, og opdelingen af disken i kataloger, filer, skjulte filer og ledig plads.

Desuden vises også den totale størrelse PC'ens RAM lager har (under 640 Kb), og hvormeget ledig RAM, der er til rådighed:

```
Volume JETTES PC created 18 Mar 1991 11.25  (a)
Volume Serial Number is 1782-49E0           (b)

 33462272 bytes total disk space            (c)
    55296 bytes in 3 hidden files           (d)
   147456 bytes in 68 directories           (e)
 29679616 bytes in 1030 user files          (f)
  3579904 bytes available on disk           (g)

    2048 bytes in each allocation unit      (h)
   16339 total allocation units on disk     (i)
    1377 available allocation units on disk (j)

  655360 bytes total memory                 (k)
  370816 bytes free                         (l)
```

Her har linierne følgende betydning:

(a) Diskens label, og dato for sidste ændring

(b) Diskens serienummer

(c) Diskens totale størrelse

(d) Skjulte filer (bl.a. **IO.SYS** og **MSDOS.SYS**)

(e) Kataloger

(f) Ordinære filer

(g) Ledig diskplads

(h) Antal bytes i hver sektor

(i) Antal sektorer på disken

(j) Antal ledige sektorer

(k) Størrelsen af det ordinære RAM lager (640 Kb)

(l) Ledig RAM lager under 640 Kb.

Kommandoen

CHKDSK A:

vil undersøge disketten i drev A:, mens kommandoen brugt uden argumenter vil undersøge det aktuelle drev.

Hvis der findes fejlbehæftede sektorer på disken, vil kommandoen vise dette, og man får en fejlmeddelelse:

```
Volume DDE PC316 created 9 May 1990 13:49
Volume Serial Number is 1782-49E0

Errors found, F parameter not specified.

1704 lost clusters found in 5 chains.
3489792 bytes diskplace
        would be freed.

33462272 bytes total disk space
  468992 bytes in 4 hidden files
  116736 bytes in 52 directories
28874752 bytes in 1164 user files
  512000 bytes available on disk

    2048 bytes in each allocation unit
   16339 total allocation units on disk
    1377 available allocation units on disk

655360 bytes total memory
370816 bytes free
```

Her fortæller kommandoen selv at eventuelle ændringer ikke vil blive skrevet til disken, da option /F (Fix errors) ikke er angivet.

Hvis man har kaldt kommandoen med option **/F**, vil man få yderligere spørgsmålet:

Convert lost clusters to files (Y/N)?

og hvis her svarer **Y**, vil de tabte kæder (chains) blive konverteret til filer med navnene: **FILE####.CHK**, hvor **####** står for **0000, 0001, 0002** og så fremdeles.

Disse filer kan man så læse med et redigeringsprogram, men højst sandsynligt vil man herefter slette dem, da de sjældent kan bruges til noget fornuftigt.

Visse programmer opdaterer på disken, før FAT tabellen opdateres. Hvis et sådant program afbrydes, vil de skrevne klynger være markerede som optagede i FAT tabellen, men tabellen er ikke blevet færdig-opdateret, og således opstår de tabte klynger typisk.

Efter lang tids brug vil alle klynger på disken blive brugt, og efterhånden som filer slettes, vil de ledige klynger ligge spredt rundt omkring på disken. Når der skal skrives en ny fil til disken, finder DOS selv de ledige klynger og sætter dem sammen til kæder.

Disse kæder kan således godt bestå af klynger, der ikke ligger samlet et sted på disken. Dette kaldes **disk fragmentering.**

Dette er i og for sig ikke et problem, men det tager lidt længere tid at bruge filerne, end hvis de ligger samlet.

CHKDSK kommandoen kan benyttes til at finde sådanne filer, ved at angive navnene på de filer, der skal undersøges, evt ved hjælp af masker:

CHKDSK C:*.*

I dette eksempel, gennemsøges rodkataloget på C: drevet for ikke-sammenhængede filer.

Denne kommando vil først skrive de sædvanlig oplysninger om disken, og herefter f.eks.:

C:\COMMAND.COM Contains 3 non-contiguous blocks.

Hvis det bliver et problem med for mange ikke-sammenhængende filer på en diskette, kan disketten kopieres til en ny-formateret f.eks. ved hjælp af kommandoen **XCOPY**.

Hvis problemet opstår på en harddisk, må man starte med at tage backup af hele harddisken, derefter formatere den (husk evt. at formattere med option **/S**, hvis det er den disk, der skal bootes fra).

Herefter kan filerne kopieres tilbage til disken ved hjælp af kommandoen **RESTORE**.

Dette er en omfattende proces, som man ikke bør kaste sig ud i, uden det er absolut nødvendigt, og man er fuldstændig fortrolig med de anvendte kommandoer.

CHKDSK har to mulige parametre: **/V** og **/F**.

Parameteren **/V** (verbose) benyttes til at få listet navnene på samtlige filer og kataloger på drevet.

Parameteren **/F** benyttes, som tidligere nævnt, til at konvertere tabte kæder til filer.

Opgave

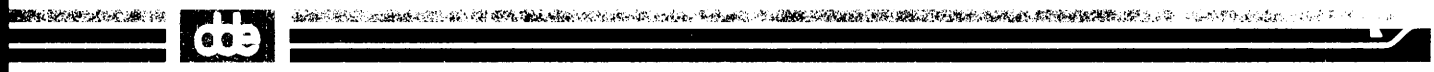
Formål: - at afprøve de mange anvendelser af CHKDSK kommandoen.

1. **Kontroller C: drevet for tabte kæder.**

2. **Konverter disse kæder til filer, og udskriv en af disse med TYPE kommandoen.**

3. **Søg C: drevet i gennem for at finde ikke-sammenhængende filer.**

4. **Skriv en batchfil, der ved hjælp af CHKDSK kommandoen finder i hvilket katalog på et givet drev, at en fil findes.**



11 ANSI.SYS

ANSI.SYS er en driver, der muliggør operationer på skærm og tastatur. Driveren indeholder en række standard rutiner til:

- at slette skærmen
- at slette linier på skærmen
- at ændre farver på skærmen
- at omdefinere taster på tastaturet
- at placere markøren på skærmen.

Driveren skal indlæses via en linie i **CONFIG.SYS** filen:

DEVICE=C:\DOS\ANSI.SYS

Denne linie skal komme før en evt. linie med

DEVICE=C:\DOS\DISPLAY.SYS ...

i **CONFIG.SYS.**

Kommandoer til **ANSI.SYS** gives ved hjælp af såkaldte "ESCAPE-koder", der består af en <ESC> (ascii kode 27) efterfulgt af en "[" (ascii kode 91)

Koder til at ændre farver på skærmen:

Escape koder af formen: <ESC>[#;#;...;#m
hvor # har følgende betydning:

Special-attributter:

0	**Nulstil til hvid forgrund, sort baggrund**
1	**Fed forgrundsfarve (monokrom fed)**
4	**Understreget**
5	**Blinkende**
7	**Omvendt farve, sort på hvid**
8	**Usynlig - sort på sort**

Forgrundsfarver:

30	**Sort**
31	**Rød**
32	**Grøn**
33	**Gul**
34	**Blå**
35	**Magenta**
36	**Cyan**
37	**Hvid**

Baggrundsfarver:

40	**Sort**
41	**Rød**
42	**Grøn**
43	**Gul**
44	**Blå**
45	**Magenta**
46	**Cyan**
47	**Hvid**

Eksempel:

Koden **<ESC>[1;37;44m** vil resultere i fed hvid forgrund på blå baggrund.

Hvis man ønsker at omprogrammere en tast på tastaturet, skal man benytte en escape sekvens af formen

<ESC>[#;#;...,#p

hvor den første # står for ASCII koden for den tast, der skal omdefineres, og de næste for de koder tasten skal sende.

Hvis man ønsker at omdefinere tasten, der sender tegnet "¤", til at sende et "$", man man sende escape sekvensen:

<ESC>[175;36p

175 er ASCII koden for ¤, og 36 er ASCII værdien for $.

Koder til at programmere tastaturet (Udvidede ASCII koder):

Funktionstaster:	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
	59	60	61	62	63	64	65	66	67	68
Shift (f11-f20)	84	85	86	87	88	89	90	91	92	93
Ctrl (f21-f30)	94	95	96	97	98	99	100	101	102	103
Alt (f31-f40)	104	105	106	107	108	109	110	111	112	113

Numeriske taster:	1	2	3	4	5	6	7	8	9	0
Alt	120	121	122	123	124	125	126	127	128	129

Bogstaver	A	B	C	D	E	F	G	H	I	J
Alt	30	48	46	32	18	33	34	35	23	36

Bogstaver	K	L	M	N	O	P	Q	R	S	T
Alt	37	38	50	49	24	25	16	19	31	21

Bogstaver	U	V	W	X	Y	Z	Æ	Ø	Å
Alt	22	47	17	45	21	44	39	40	26

Disse udvidede ASCII koder sendes på samme måde som de øvrige, dog skal man angive, at der er tale om en udvidet kode ved at skrive et 0 (nul) som første tegn efter [.

Eksempel:

Hvis f10 skal sende kommandoen **DIR /P**, kan man sende følgende escape sekvens:

<ESC>[0;68;"dir /p";13p

hvor 68 er den udvidede ASCII kode for f10 tasten, og 13 er ASCII koden for **<Retur>**.

Da <ESC> tasten har en specialbetydning ved DOS prompten (Fortryd indtastning, og start forfra), kan man ikke angive en escape-sekvens her.

Disse kan angives på 2 forskellinge måder:

Den letteste er at benytte **PROMPT** kommandoen. Her findes et specialtegn **$e**, der netop står for Escape.

Eksemplet fra før, med hvid tekst på blå skærm, kan altså skrives:

```
PROMPT $E[1;44;37m$P$G
```

Hver gang man herefter taster retur, vil escapesekvensen blive sendt, og dette er ikke særligt performance venligt, så man bør herefter afgive kommandoen

```
PROMPT $P$G
```

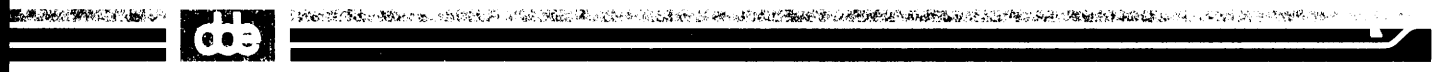
der giver den sædvanlige prompt, men farverne på skærmen vil fortsat være, som sat i den foregående kommando.

Den anden mulighed er at skrive escape-sekvensen ved hjælp af en editor, der kan håndtere sådanne specialtegn. Man kan herefter sende escape-sekvensen ved at udskrive filen med **TYPE** kommandoen.

Til dette formål kan man benytte DOS udgaven af **vi** editoren. Når man er kommet ind i **vi** editoren i skrive-tilstand (ved kommandoen i, I, o, O, a, A, ..), kan man vise, at det næste tegn er et specialtegn ved at skrive **<CTRL+V>** efterfulgt af det ønskede tegn.

Et **<ESC>** vil således blive skrevet på skærmen som "^[", og den før nævnte escape sekvens vil komme til at se således ud på skærmen:

```
^[[1;37;44m
```



Bilag 1

ASCII TABELLEN

	0	1	2	3	4	5	6	7	8	9
0x	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1x	nl	vt	ff	cr	so	si	dle	dc1	dc2	dc3
2x	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3x	rs	us	sp	!	"	#	$	%	&	'
4x	(	)	*	+	,	-	.	/	0	1
5x	2	3	4	5	6	7	8	9	:	;
6x	<	=	>	?		A	B	C	D	E
7x	F	G	H	I	J	K	L	M	N	O
8x	P	Q	R	S	T	U	V	W	X	Y
9x	Z	[	\	]	^	-	`	a	b	c
10x	d	e	f	g	h	i	j	k	l	m
11x	n	o	p	q	r	s	t	u	v	w
12x	x	y	z	{	¦	}	~	del		

Opgave

Formål: - at omdefinere skærmen og tastaturet.

1. **Lav de ændringer, der er beskrevet i teksten, ved hjælp af PROMPT kommandoen.**

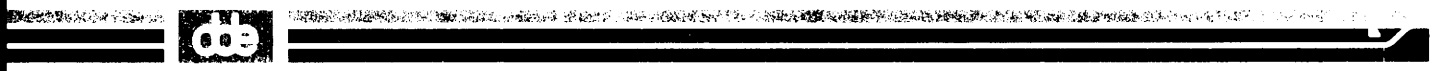
2. **Start en ny kommandofortolker ved at afgive kommandoen COMMAND. Gælder ændringerne stadig?**

3. **Gå tilbage til den forrige shell ved at taste EXIT, og se om ændringerne gælder her.**

4. **Indsæt en kommando i AUTOEXEC.BAT på systemdisketten, der sætter skærmens baggrundsfarve til rød og teksten til hvid.**

   Boot, så du kan se ændringerne i funktion.

5. **Benyt nu vi-editoren til af omdefinere tasten med # til at sende en \ (backslash), og f9 til at sende ¦ (pipetegn).**



12 Kopiering

Kopiering af filer i **DOS** kan ske ved hjælp af kommandoerne **COPY**, **XCOPY** og **BACKUP/RESTORE**.

Der findes endnu en kopierings kommando, **DISKCOPY**, i DOS, der kopierer en hel diskette, bit for bit til en anden diskette af eksakt samme format.

12.1 DISKCOPY

Syntaxen for kommandoen er:

DISKCOPY A: B:

Denne kommando vil kopiere disketten i drev A: til en diskette i drev B:, hvis de to disketter er af samme format.

Hvis man kun har et diskettedrev i det pågældende format, kan man afgive kommandoen

DISKCOPY A: A:

og man vil så blive anmodet om at skifte diskette under vejs, idet der læses så meget ind i RAM lageret ad gangen, som der er plads til. Man kan blive bedt om at skifte diskette mange gange før kopieringen er afsluttet.

Forskellen mellem kommandoerne **COPY/XCOPY** og **DISKCOPY** er, at med **DISKCOPY** kommandoen kopieres filerne til præcis det samme cluster på den nye diskette, og skjulte filer kopieres med. **COPY/XCOPY** kopierer bare til første ledige cluster.

Således kan **DISKCOPY** bruges til at kopiere systemdisketter, hvor de to skjulte filer **IO.SYS** og **MSDOS.SYS** skal være placeret, så **IO.SYS** ligger placeret på de første sektorer, der er afsat til filer, og **MSDOS.SYS** følger i umiddelbar fortsættelse af denne.

Kommandoen kan også benyttes til at kopiere til disketter, der ikke er formatterede, da hele bitmønstret fra den oprindelige diskette kopieres til den nye diskette.

Disketten formateres således under vejs.

DISKCOPY er ikke egnet til at lave sikkerhedskopiering med, bl.a. da kommandoen ikke ændrer ved arkivbitten.

12.2 DISKCOMP

Kommandoen

DISKCOMP A: B:

kan bruges til at sammenligne to disketter af samme format.

Selv om to disketter indeholder de samme filer, kan filerne ligge forskelligt placeret på de to disketter, og kommandoen **DISKCOMP** vil i så tilfælde fortælle, at de to disketter er forskellige

Typisk vil man benytte **DISKCOMP** lige efter en **DISKCOPY** til at kontrollere, at kopieringen gik godt.

Alternativt kan man før kopieringen afgive kommandoen

VERIFY ON

der kontrollerer samtlige skrivninger til den nye diskette samtidig med kopieringen. Kopieringen vil således tage lidt længere tid.

12.3 COPY

COPY kommandoen har en kommando til kontrol af det kopierede svarende til **DISKCOMP**, nemlig **COMP** kommandoen.

Denne kommando kontrollerer at to filer er ens.

En **COPY** kommando efterfulgt af en **COMP** kommando, vil svare til en **COPY** kommando med option **/V**, der verificerer kopieringen, eller til som før at skrive

VERIFY ON

inden kopieringen.

COMP kommandoen vil først sammenligne størrelsen af de to filer, og herefter vil den rapportere de 10 første forskelle mellem de to filer.

Opgave

Formål: - at afprøve kommandoerne DISKCOPY, DISKCOMP og VERIFY ON.

1. **Lav en DISKCOPY af systemdisketten til øvedisketten.**

   Hvad ligger der nu på øvedisketten?

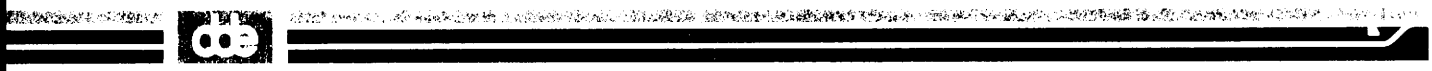
   Kan man boote fra øvedisketten?

2. **Kontroller at de to disketter er helt ens med DISKCOMP kommandoen.**

3. **Formatter nu øvedisketten, og kopier alt fra systemdisketten til øvedisketten ved hjælp af XCOPY kommandoen.**

   Kontroller nu ved hjælp af DISKCOMP om de to disketter stadig er ens.

4. **Slå nu verify til, og lav igen en diskkopi af systemdisketten til øvedisketten.**



13 Backup procedurer

Det er meget væsentligt at sørge for at sikkerhedskopiere jævnligt. Både på grund af faren for hardware fejl, men også fordi man selv meget let kan komme til at slette filer ved en fejltagelse.

DOS har kommandoen **BACKUP**, der med sine mange options kan benyttes til at udføre denne sikkerhedskopiering.

Selve backup proceduren kan man "pakke ind" i en batchfil, så brugeren ikke behøver huske på hvilke options, der skal med, og så kan man lægge et kald af denne ind i en menu, eller oprette en ikon til det i Windows.

Det eneste brugeren så behøver tænke på er, hvor mange formaterede disketter, der skal benyttes, til dagens eller ugens backup.

Hvis man vil lave en total backup af et drev, kan man få et overblik over antallet af disketter, der skal bruges, ved at benytte kommandoen **CHKDSK**.

Denne kommando giver det totale antal bytes og antallet af frie bytes. Antallet af disketter kan man så beregne ved:

$$\textbf{Antal disketter} = \frac{\textbf{(total bytes - frie bytes)}}{\textbf{bytes på disketten}}$$

Hvis man benytter 1,44 Mb disketter (1.457.664 bytes), det totale antal bytes er 33.462.272 (32Mb), og antallet af frie bytes er 3.497.984, ser regnestykket således ud:

$$\frac{33.462.272-3.497.984}{1.457.664} = 20,55$$

så her skal der altså benyttes 21 disketter til en total backup.

Man kan ikke på tilsvarende måde beregne antallet af disketter, der skal benyttes til en tilvækstbackup.

Når man skal udføre en total backup, skal man anvende option /S til backupkommandoen for at få taget samtlige underkataloger med.

En tilvækstbackup tager man ved at benytte option /S/A/M for at få taget underkataloger med (/S), for at tilføje til forrige backup (/A) og for kun at få taget de filer med, der er blevet redigeret siden sidste backup (/M).

En total backup af C: drevet ser således ud:

```
BACKUP C:\*.* A: /S
```

og en tilvækstbackup ser således ud:

```
BACKUP C:\*.* A: /S/A/M
```

En backup procedure kan være, at man f.eks. én gang om måneden tager en total backup og herefter tager en tilvækst backup hver dag.

Enhver backup procedure vil i alle tilfælde resultere i et væld af disketter. For at kunne holde rede på disse er det nødvendigt at man mærker dem omhyggeligt.

Man kan desuden benytte sig af, at **BACKUP** kommandoen fortæller hvilke filer, der tages backup af.

Dette output kan man enten sende videre til en printer eller gemme i en logfil, således at en total-backup overskriver den gamle logfil og en tilvækst-backup tilføjer til den gamle logfil.

Man vil da til hver en tid kunne se, på hvilken diskette sidste kopi af en given fil befinder sig, hvis det bliver nødvendigt at rekonstruere den.

13.1 Oprydning efter diskfragmentering

Hvis harddisken efter lang tids brug er blevet meget fragmenteret, kan man tage en komplet backup af den, formatere den og derefter rekonstruere filerne fra backuppen.

Det er en omfattende procedure, som man ikke skal udføre, før det er absolut nødvendigt.

Proceduren går ud på at tage backup af disken, og af hensyn til diskettefejl, tages endnu en backup.

Disse to backup'er kan fylde op til 2 gange 23 1,44 Mb disketter, som skal være formaterede, inden man starter.

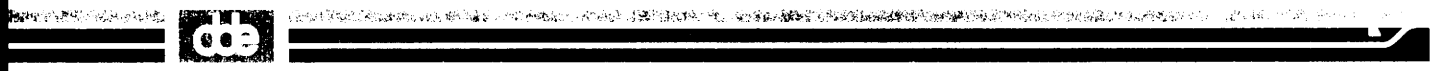
Proceduren kan udføres ved hjælp af følgende batchfil:

```
ECHO OFF
CLS
REM DISK FRAGMENTATIONS PROCEDURE

BACKUP C:\ A: /S

IF ERRORLEVEL 4 GOTO FEJL
IF ERRORLEVEL 3 GOTO BRUGER-STOP
IF ERRORLEVEL 2 GOTO FEJL
IF ERRORLEVEL 1 GOTO INGEN-FILER

ECHO Første backup afsluttet, nu begynder næste backup
PAUSE

BACKUP C:\ A: /S

IF ERRORLEVEL 4 GOTO FEJL
IF ERRORLEVEL 3 GOTO BRUGER-STOP
IF ERRORLEVEL 2 GOTO FEJL
IF ERRORLEVEL 1 GOTO INGEN-FILER
```

```
ECHO Anden backup afsluttet, nu formateres harddisken!
PAUSE
FORMAT C:

ECHO Nu genlagres filerne fra backuppen.
RESTORE A: C:\ /S

IF ERRORLEVEL 4 GOTO R-FEJL
IF ERRORLEVEL 3 GOTO R-BRUGER-STOP
IF ERRORLEVEL 2 GOTO R-FEJL
IF ERRORLEVEL 1 GOTO R-INGEN-FILER

ECHO Det var så det!

GOTO DONE

:INGEN_FILER
ECHO Der var ingen filer at tage backup af.
GOTO DONE

:FEJL
ECHO Der skete fejl under backuppen.
GOTO DONE

:BRUGER-STOP
ECHO Du har stoppet mig midt i backuppen.
GOTO DONE

:R-FEJL
ECHO Der skete fejl under genlagringen.
GOTO DONE

:R-INGEN-FILER
ECHO Der var ingen filer at genlagre.
GOTO DONE

:R-BRUGER-STOP
ECHO Du har stoppet mig midt i genlagringen.
GOTO DONE

:DONE
```

Her benyttes de exitkoder som hhv. backup og restore giver til at skrive passende meddelelser til brugerne.

Flere af de benyttede labels er længere end 8 tegn, men kun de første 8 tegn evalueres.



Opgave

Formål: - at udvikle en batchfil, der udfører backup.

1. **Lav en lille batchfil, der udfører en total backup af C: drevet.**

2. **Lav en batchfil, der udfører en tilvækstbackup af C: drevet.**

3. **Lav nu en batchfil, der som argument tager total eller tilv, og så udfører denne form for backup.**

   Sørg for at der udskrives gode vejledninger til brugeren.

14 DOS' Virkemåde

DOS er inddelt i tre sektioner: Kommandofortolkeren, kernen og BIOS. Kommandofortolkeren behandler de kommandoer, man afgiver. Kernen står for kommunikationen mellem brugeren og DOS, og BIOS står for al input og output.

14.1 Kommandofortolkeren

Standard kommandofortolkeren i DOS er **COMMAND.COM**. Man kan dog definere sin egen kommandofortolker ved at tildele variablen **COMSPEC** en anden værdi i **CONFIG.SYS** filen.

DOS kommandofortolkeren består af tre dele: en opstarts del, en resident del, og en transient del.

Opstartsdelen's eneste funktion er at søge efter filen **\AUTOEXEC.BAT** ved boot og udføre den, hvis den eksisterer. Når dette er overstået, frigives den plads i RAM, som blev benyttet af denne del.

Den residente del af kommandofortolkeren forbliver i RAM hele tiden. Denne del klarer alt det, som DOS skal kunne give øjeblikkelig reaktion på: loadning af den transiente del, afslutte udførelse af programmer eller bruger-interrupt (<Ctrl+C>).

Desuden tager den residente del sig af den kritiske fejlhåndtering. Det er bl.a. denne del, der er ansvarlig for fejlmeddelelsen:

Not ready error reading device
Abort, Retry, Ignore?

Den tredie del af kommandofortolkeren, den transiente del, ligger ikke konstant i RAM lageret. Mange applikationer overskriver denne del. Når en sådan applikation terminerer, og der ikke kan findes en **COMMAND.COM** fil, vil man få en fejlmeddelelse:

Insert disk with COMMAND.COM in drive A
and strike any key when ready

Her skal man blot indsætte en diskette med **COMMAND.COM** og trykke på en tilfældig tast.

Den transiente del modtager kommandoer fra brugeren eller fra batchfiler, og er ansvarlig for udførelsen.

De interne DOS kommandoer er indeholdt i denne del:

BREAK	**CHDIR**	**CLS**	**COPY**	**CTTY**
DATE	**DEL**	**DIR**	**ERASE**	**MD**
PATH	**PROMPT**	**REN**	**RMDIR**	**SET**
TIME	**TYPE**	**VER**	**VERIFY**	**VOL**

Hvis en kommando, der afgives, ikke er en intern DOS kommando, søges efter en fil i det aktuelle katalog med det pågældende navn og filtypen **.COM**, **.EXE** eller **.BAT**.

Hvis en sådan ikke findes, søges efter filen i hvert af de kataloger, der er angivet i brugerens **PATH**.

Hvis det heller ikke giver resultat, vil man få følgende fejlmeddelelse:

Bad command or file name

14.2 DOS kernen

Kernen af DOS er ansvarlig for:

fil administration
(oprettelse, sletning eller redigering af DOS filer)

katalog administration
(oprettelse, sletning eller ændring af kataloger)

applikations interface til DOS services.

Ved boot læses filen **MSDOS.SYS** ind i RAM lageret for at oprette DOS kernen.

DOS kernen skaber forbindelse mellem brugerens programmer og DOS, således at når et program skriver til skærmen, disken eller printeren, så benyttes de services, som DOS kernen tilbyder.

14.3 BIOS

Til enhver PC findes et sæt af rutiner til at udføre input og output på laveste niveau.

Disse rutiner befinder sig i den del af lageret, der kaldes **Read Only Memory**, eller **ROM lageret.**

Denne del af lageret bliver i modsætning til **RAM** lageret ikke slettet ved boot, og man kan heller ikke skrive i **ROM** lageret.

ROM lageret består dels af nogle boot rutiner, dels nogle **BIOS** rutiner. (**Basic Input Output System**)

Hver gang PC'en bootes, benytter DOS de BIOS rutiner, der findes i ROM lageret sammen med filen **IO.SYS** til at oprette den del af lageret, der skal være ansvarlig for input og output.

DOS kommunikerer med disse rutiner ved at udveksle pakker med information.

I mange tilfælde kræves en såkaldt device driver for at oversætte den information fra et format, som DOS forstår, til et format, som den ydre enhed forstår.

Device drivers gør det let at installere nyt hardware. Man skal blot installere en driver til det pågældende i **CONFIG.SYS**:

```
DEVICE=C:\DOS\HIMEM.SYS
```

installerer driveren **HIMEM.SYS**, der gør det mulig at anvende det RAM lager, der ligger mellem 640 Kb og 1 Mb.

14.4 Boot sektoren

Bootsektoren, første sektor på en disk eller diskette, skal findes på enhver DOS disk. Denne sikrer, at DOS systemet kan loades fra en systemdiskette, og at meddelelsen:

Non-system disk og disk error
Replace and strike any key when ready

fremkommer, hver gang man forsøger at boote fra en ikke-systemdiskette.

Når man booter, vil DOS først søge efter en systemdiskette i drev A: herefter i drev B: og først herefter på harddisken C:.

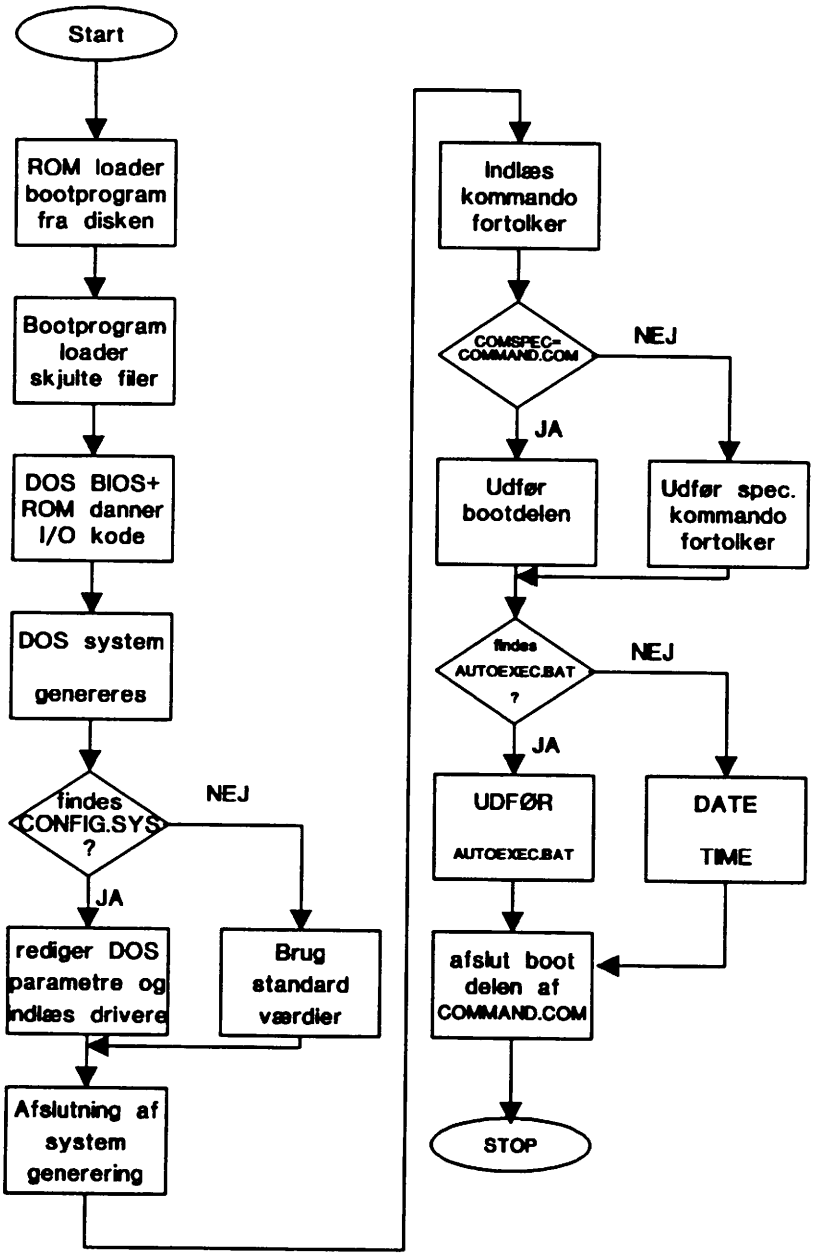
Dette er en sikkerhed for brugeren, for hvis harddisken er blevet beskadiget, kan man stadig boote PC'en ved at indsætte en diskette i et af drevene og boote fra den.

14.5 Systemgenerering

Når man booter, sørger boot-delen af ROM lageret for at boot programmet fra boot sektoren på disken læses ind i RAM lageret.

Herefter loades filerne **MSDOS.SYS** og **IO.SYS** (eller i visse versioner **IBMDOS.COM** og **IBMBIO.COM**) fra disken ind i RAM, boot programmet terminerer og overlader kontrollen til DOS.

Denne procedure kan ses på næste side.


Start
ROM loader bootprogram fra disken
Bootprogram loader skjulte filer
DOS BIOS+ ROM danner I/O kode
DOS system genereres
findes CONFIG.SYS ?
NEJ
JA
rediger DOS parametre og indlæs drivere
Brug standard værdier
Afslutning af system generering
Indlæs kommando fortolker
COMSPEC=COMMAND.COM
NEJ
JA
Udfør bootdelen
Udfør spec. kommando fortolker
findes AUTOEXEC.BAT ?
NEJ
JA
UDFØR AUTOEXEC.BAT
DATE
TIME
afslut boot delen af COMMAND.COM
STOP

BOOT MED DOS

14.6 Interrupts

Et interrupt er et signal til CPU'en fra et program eller en ydre enhed. Et interrupt vil midlertidigt afbryde CPU'en og få den til at udføre en anden opgave.

Hver interrupt har en stump kode liggende i RAM lageret, som processoren udfører hver gang interruptet forekommer.

Denne stump kode kaldes en interrupt handler eller en interrupt rutine.

Et eksempel på et interrupt er, hvis brugeren taster <PrtSc>. De processer, der kører, standses midlertidigt, mens skærmbilledet skrives til printeren.

Når processoren er færdig med dette, vender den tilbage til de afbrudte processer.

CPU'en finder den rigtige interrupt-rutine ved hjælp af en interrupt-vektor.

En interrupt-vektor er en indgang i en tabel, hvor adressen på den stump kode, der skal udføres, er angivet.

15 DEBUG

DEBUG kommandoen er et program, der kan benyttes til fejlsøgning eller redigering i eksisterende programmer.

Kommandoen startes ved at skrive:

```
DEBUG
```

Kommandoens klarsignal er et

```
-
```

Hvis man f.eks gerne vil undersøge hvilken udgave af ROM BIOS man har, kan dette aflæses på lagerets allerhøjeste adresse:

```
-d ffff:0000 L 10
```

Her vil man på skærmen kunne aflæse de højeste 16 bytes i lageret, og BIOS datoen kan direkte aflæses:

```
FFFF:0000  EA 5B E0 00 F0 30 36 2F-31 39 2F 39 30 00 FC F9
                                          .[...06/19/90...
```

Datoen er ikke den dag, hvor PC'en er fremstillet, men datoen for BIOS.

Den næstsidste byte viser hvilken PC type, der er tale om.

FF	**Den oprindelige PC**
FE	**PC/XT, og den transportable PC**
FD	**PCjr, som nok næppe findes i Danmark**
FC	**Alle 80286 baserede PC'ere og PS/2**
FB	**PC/XT, modellerne SFD og SDD**
FA	**PS/2, model 30**
F9	**Convertible PC**

Mange programmer benytter denne bit til at teste, hvilken type PC det er.

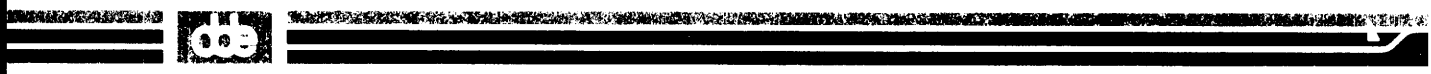
Følgende eksempel viser endnu en anvendelse af **DEBUG** kommandoen, nemlig hvordan man kan lave et program, der udfører en system reset (svarende til **<Ctrl+Alt+Del>**):

```
-a                                          assemble
2DB5:0100 mov dx,40
2DB5:0103 mov ds,dx
2DB5:0105 mov bx,72
2DB5:0108 mov word ptr[bx],1234
2DB5:010C jmp ffff:0
2DB5:0111 <Retur>

-r cx                                       aflæs indhold af cx
CX 0000
:11                                         sæt cx til 11 (hex)

-n boot.com                                 giv programmet et
                                            navn
-w                                          skriv til disk

Writing 11 bytes

-q                                          quit
```

Dette program foretager følgende: på adressen 0040:0072 skal konstanten 1234 placeres, og herefter skal der foretages et hop til adressen ffff:0000.

Registret cx, angiver størrelsen af programmet, og sættes her til 17 decimalt, som svarer til 11 hexadecimalt. Dette tal kan aflæses af linienummereringen.

Når man herefter taster **BOOT** vil PC'en udføre en system reset.



Dansk Data Elektronik A/S
Herlev Hovedgade 199
DK 2730 Herlev
Tel.: (+ 45) 42 84 50 11
Fax: (+ 45) 42 84 52 20

DDE-Subsidiaries:

BELGIUM
DDE Belgium N.V. - Excelsiorlaan 45, B8. - B1930 Zaventem - Belgium
Tel.: (+ 32) 2 725 12 25 - Fax: (+ 32) 2 726 03 05

■ GREAT BRITAIN
DDE Great Britain Ltd. - Rosemount House, Rosemount Avenue. - West Byfleet, Surrey KT14 6NP - Great Britain
Tel.: (+ 44) 932 336011 - Fax: (+ 44) 932 336603

■ NEW ZEALAND
Dansk Data Elektronik (NZ) Ltd. - 598 Main Street. - Palmerston North - New Zealand
Tel.: (+ 64) 63 61544 - Fax: (+ 64) 63 71522

NORWAY
DDE Norge A.S. - Postbox 3219, Elsenberg. - 0208 Oslo 2. - Norway
Tel.: (+ 47) 2 831155 - Fax: (+ 47) 2 830954

■ SPAIN
Dansk Data Elektronik S.A. - Entenza 332-334, 7°, 2ª. - E-08029 Barcelona. - Spain
Tel.: (+ 34) 3 4191836 - Fax: (+ 34) 3 3228804

■ SWEDEN
DDE Sverige AB. - Kanalvägen 12. - S-194 61 Upplands Väsby - Sweden
Tel.: (+ 46) 760 74040 - Fax: (+ 46) 760 74485