9. Schijfbeheer

9.1 Inleiding

Het is nog helemaal niet zo erg lang geleden dat het bezit van een harde schijf in de computer thuis, een bijna niet te verwezenlijken ideaal was. Die waren zo duur dat de gemiddelde computergebruiker het zich niet kon permitteren er een aan te schaffen.

Inmiddels is er veel veranderd. Voor een fractie van de prijs van toen, kunnen we nu harde schijven kopen met een onvoorstelbaar grote capaciteit. Praktisch iedere PC is nu met een harde schijf uitgerust. Diskettes worden echter ook nog steeds gebruikt. De programma's die je koopt staan op diskettes, gegevensuitwisseling vindt plaats via diskettes en diskettes worden gebruikt als drager van gegevens.

De opslagcapaciteit van harde schijven en diskettes wordt uitgedrukt in kilobytes of in megabytes. Een kilobyte is 1024 bytes (2 tot de macht 10). Een megabyte is 1024 maal 1024 bytes, ofwel 1.048.576 bytes. Kilobyte wordt afgekort tot Kb en megabyte tot Mb. Harde schijven worden in een grote verscheidenheid van opslagcapaciteit op de markt aangeboden. Een opslagcapaciteit van 250 megabyte is al niet meer zo erg bijzonder.

Diskettes zijn er in de formaten 3½ en 5¼ inch. De 5¼-inch diskette is de wat ouderwets aandoende slappe schijf. Deze diskettes zijn in Double Density (DD) en in High Density (HD) verkrijgbaar. Double Density diskettes hebben een opslagcapaciteit van 360 kilobyte en High Density van 1,2 megabyte. Alleen computers van minstens de AT-klasse kunnen 1,2 megabyte-diskettes verwerken.

De 3½-inch diskette is de diskette met het harde plastic omhulsel. Deze diskette kan in de Double Density-uitvoering 720 kilobyte bevatten en in High Density 1,44 megabyte. Er zijn zelfs machines die speciale 3½-inch diskettes een capaciteit van 2,88 megabyte kunnen geven.

Een diskette wordt aan twee zijden gebruikt. Een harde schijf bestaat meestal uit een pakket boven elkaar liggende schijven. Behalve voor de buitenste twee kanten, is er voor iedere zijde van een schijf een lees- en schrijfkop beschikbaar.

Door het besturingssysteem worden de schijven zodanig ingedeeld dat ermee gewerkt kan worden. Dit indelen van de schijven wordt formatteren genoemd. Tijdens het formatteren worden op iedere schijfkant cirkels getrokken. Deze cirkels worden sporen genoemd. Ieder spoor is verdeeld in een aantal sectoren. In iedere sector kan een bepaald aantal bytes geschreven worden.

Een 360 Kb-diskette heeft aan weerszijden 40 sporen. Ieder spoor bevat 9 sectoren. In iedere sector kunnen 512 bytes weggeschreven worden. Met deze gegevens kunnen we berekenen hoeveel bytes er op een diskette passen:

2 zijden x 40 sporen x 9 sectoren x 512 bytes = 368640 bytes

Dit berekende getal komt overeen met de uitkomst van 360 x 1024 bytes.

Boven elkaar liggende sporen op een diskette of harde schijf worden cilinders genoemd.

Het besturingssysteem beheert de schijf als een administratie waarin vastgelegd is welke clusters op de schijf bezet zijn. Een cluster bevat, afhankelijk van het type schijf, één of meer sectoren. In het voorbeeld van de 360 Kb-diskette is een cluster twee sectoren groot, ofwel 1024 bytes. Elke cluster op de schijf heeft een nummer. Deze manier van administreren heeft de consequentie dat bestanden op schijf weggezet worden in clusters van 1024 bytes. Ook al is een naar de schijf te schrijven bestand maar 1 byte groot, het neemt toch 1024 bytes in beslag.

Iedere geformatteerde schijf heeft een zogenaamde root directory. Deze directory kun je het beste zien als een inhoudsopgave van de bestanden die de schijf bevat. Voor ieder bestand is er een record van 32 bytes beschikbaar. Dit record bevat informatie over het bestand. Zo'n record is als volgt opgebouwd:

Bestandsnaam : 8 bytes

```
Bestandskenmerk  :  3  bytes
Attributen       :  1  byte
Gereserveerd     : 10  bytes
Tijd             :  2  bytes
Datum            :  2  bytes
Eerste Cluster   :  2  bytes
Grootte bestand  :  4  bytes
                    ===
     Totaal          : 32  bytes
```

Hier kunnen we zien dat de naam van een bestand maximaal acht lettertekens groot mag zijn. Een bestandsnaam kan gevolgd worden door een kenmerk van maximaal drie letters. Dit kenmerk wordt ook wel de extensie genoemd. Door het besturingssysteem worden bestandsnaam en extensie van elkaar gescheiden door een punt.

Attributen worden gebruikt om door middel van een bitpatroon vast te leggen of het om een systeemfile gaat, of er in het bestand geschreven mag worden, of het verborgen moet worden, enzovoorts.

In Datum en Tijd worden de datum en het tijdstip gezet waarop het bestand de laatste keer naar de schijf geschreven is. In het laatste veld wordt bijgehouden hoeveel bytes het bestand omvat.

In het voorlaatste veld staat het nummer van de eerste cluster op de schijf waar het bestand in gezet is. Dit clusternummer wordt opgezocht in de FAT. Dit is een afkorting van "File Allocation Table". Deze tabel bevat een administratie van alle clusters op schijf. In het bestandsrecord staat het nummer van de eerste cluster. Als deze cluster wordt opgezocht in de FAT, dan vinden we hier het nummer van de volgende cluster. In dat volgende cluster kunnen we weer een clusternummer van het daarop volgende cluster zoeken. In het laatste cluster staat het hexadecimale getal FFFF, ten teken dat het einde van het bestand bereikt is.

Het zal duidelijk zijn dat een bestand niet noodzakelijkerwijs als één geheel op de schijf gezet wordt. In welke clusters het bestand geplaatst wordt, is afhankelijk van de vraag welke clusters op dat moment vrij zijn.

Nog even een voorbeeldje. Stel dat we een bestand van 2875 bytes naar schijf willen schrijven. Het besturingssysteem zet de eerste 1024 bytes in de eerste vrije cluster. Laten we aannemen dat dit cluster nummer 10 is. In het bestandsrecord wordt in het veld Eerste Cluster de waarde 10 genoteerd. Als de volgende vrije cluster nummer 25 is, dan worden daar de volgende 1024 bytes geplaatst. In de FAT wordt in cluster 10 een 25 genoteerd. We moeten nu nog 827 bytes kwijt. Deze worden in cluster 50 geschreven. In cluster 25 wordt naar cluster 50 verwezen en cluster 50 verwijst naar FFFF, het laatste cluster. Het feitelijke ruimtebeslag van de 2875 bytes wordt dan: 3 x 1024 = 3072 bytes. Op de volgende pagina staat een voorstelling van deze clusters met verwijzingen naar volgende clusters en een overzicht van de directory-structuur.

Bestandsrecord

(25 (((((((((((
((((((((((
FAT - Cluster 25 (
(((((((((<((((((((((
(50 (((((((((((
((((((((((
FAT - Cluster 50 (
((((((((((
(FFFF (<(((((((((
(((((((((
FAT - Laatste cluster

Directory-structuur

((((((((((((((((((
(Bestand_1 (
(Bestand_2 (Root directory
(Subdirectory_1 (
(Subdirectory_2 (
((((((((((((((((((
((((((((((((((((((
(((((((((((((((((((((((((((((((
(Subdirectory_3 ((Subdirectory_4 (Subdirectory
(((((((((((((((((((((((((((((((
((
(((((((((((((((((((((((((((((((
(Bestand_3 ((Bestand_5 (
(Bestand_4 ((Bestand_6 (Subdirectory
(Subdirectory_5 (((
(((((((((((((((((((((((((((((((

Iedere schijf heeft altijd slechts één root directory. Hierin kunnen bestandsnamen geschreven worden, maar ook zogenaamde subdirectories. In subdirectories kunnen ook weer inhoudsopgaven van bestanden gemaakt worden, maar ook nieuwe subdirectories. Deze laatste kunnen op hun beurt weer gebruikt worden voor bestandslijsten maar ook weer voor subdirectories.

De weg door de directories naar het bestand wordt een pad genoemd. Ook het Engelse woord "path" wordt vaak gebruikt. Stel dat bovenstaande directory-structuur zich op schijf C: bevindt en dat er zich in Subdirectory_3 een Bestand_4 bevindt. Het pad wordt dan:

C:\Subdirectory_1\Subdirectory_3\Bestand_4

Na de vermelding van de drive volgt een achteroverliggende schuine streep (backslash), met daarna een verwijzing naar de eerste subdirectory. De eerste backslash is de aanduiding voor de root directory. Na de eerste backslah worden andere backslashes gebruikt om de directories van elkaar te scheiden. Achter de

laatste backslash wordt de bestandsnaam opgegeven.

MS-DOS heeft altijd een actieve (current) directory. Dit is de directory waar het besturingssysteem als eerste in kijkt om een programma te starten of om gegevens in te lezen. Je kunt zo'n actieve directory zelf instellen. Hoe dit gaat, staat in de volgende paragrafen.

9.2 Zoeken van een bestand

Turbo Pascal beschikt over een groot aantal procedures en functies om de schijf vanuit je programma te beheren. Deze procedures en functies zijn ondergebracht in de unit DOS. Ze zijn toegankelijk als in je programma de opdracht USES DOS is opgenomen.

Het programma SCHIJF_1 zoekt naar een bestand in de actieve directory. Als dit niet gevonden wordt, dan wordt een nieuw bestand op schijf aangemaakt en wordt daar melding van gemaakt. Als het bestand wel gevonden wordt, dan wordt de grootte van het bestand gerapporteerd:

```
PROGRAM SCHIJF_1;
USES CRT, DOS;

VAR
   F      : File;
   REC    : SearchRec;
   BESTAND: PathStr;

BEGIN
   ClrScr;
   BESTAND := 'TEST.DAT';
   Assign(F,BESTAND);
   FindFirst(BESTAND,ReadOnly,REC);
   IF DOSError <> 0 THEN
   BEGIN
      Rewrite(F);
      Writeln('Nieuw bestand gemaakt')
   END
      ELSE
   BEGIN
      Reset(F);
      WITH REC DO
      Writeln('Bestand gevonden en geopend:',Name,
      '  = ', Size,' bytes groot.')
   END;
   Readln;
   Close(F)
END.
```

Regels:Toelichting:

[1] 3-6Declareer de benodigde variabelen.

```
   [2]9-10Geef de variabele BESTAND de waarde 'TEST.DAT' en
          verbind deze met de variabele F die van het type File
          is.
   [3]11Roep de procedure FindFirst aan.
   [4]12-23Als DOSError ongelijk is aan 0, schrijf dan een nieuw
          bestand naar de schijf. Als DOSError 0 is, schrijf dan
          uit de variabele REC de naam en de afmeting van het
          bestand naar het scherm.
25        Sluit F.
```

Toelichting:

[1]Bij de declaratie van de variabelen staan twee onbekende typen: SearchRec en PathStr. PathStr is in de DOS-unit gedefinieerd als String[79]. In DOS zijn meer van dit soort typen gedefinieerd:

```
|Type:                      |Bestemming:                        |
|ComStr : String[127]       |Commando voor DOS.                 |
|PathStr: String[79]        |Pad-aanduiding.                    |
|DirStr : String[67]        |Vastleggen directorynaam.          |
|NameStr: String[8]         |Bestandsnaam.                      |
|ExtStr : String[4]         |Extensie van het bestand.          |
```

Ook SearchRec is in de unit DOS gedefinieerd. SearchRec wordt als VAR-parameter naar de procedure FindFirst gestuurd. FindFirst zet er de gegevens van een gevonden bestand in. In de unit DOS is SearchRec als volgt gedefinieerd:

SearchRec = Record

```
   Fill: Array[1..2] OF Byte;
   Attr: Byte;
   Time: Longint;
   Size: Longint;
   Name: String[12];
     END;
```

Het veld Fill wordt door DOS gebruikt. Attr bevat de wijze waarop de attribuutbits van het bestand gezet zijn. In Time staan de datum en de tijd. In Size staat de grootte en in Name de naam van het bestand.

[2]Aan de variabele BESTAND geven we de waarde "TEST.DAT".

Deze bestandsnaam wordt met behulp van Assign verbonden met de ongetypeerde file F.

[3]FindFirst wordt aangeroepen en krijgt als parameters het pad en een attribuutwaarde waarnaar gezocht moet worden mee. Deze bestandsattribuutwaarden zijn in de unit DOS als constanten gedeclareerd:

Constante:	Waarde:	Betekenis:
ReadOnly	$01	Naar bestand kan niet geschreven worden, en het bestand kan niet worden verwijderd.
Hidden	$02	Bestand houdt zich verborgen.
SysFile	$04	Bestand is een systeembestand.
VolumeID	$08	Bestand is naam van de schijf.
Directory	$10	Bestand is een directory.
Archive	$20	Wordt gezet als het bestand gewijzigd wordt.
AnyFile	$3F	Alle bestanden.

FindFirst zoekt het eerste gewone bestand en/of het eerste bestand waarvan de attribuutwaarde overeenkomt met de doorgegeven attribuutparameter.

[4]Om te controleren of een operatie succesvol is verlopen, is in de unit DOS de variabele DOSError gedeclareerd. Als een bestand gevonden is, staat DOSError op 0. Is er een verkeerd pad opgegegeven, dan staat DosError op 3. Als geen bestand gevonden is, geeft DOSError 18 aan. Als DOSError ongelijk is aan 0, dan is er dus geen bestand gevonden. In dat geval schrijven we de variabele F naar schijf. Als DOSError 0 is, bevat REC, die we als VAR-parameter doorgaven aan FindFirst, de gegevens van het gevonden bestand. Deze gegevens tonen we op het scherm.

9.3 Bestandenlijst zoeken

Het zal regelmatig voorkomen dat je in een directory met bestanden met dezelfde extensies of gedeeltelijk dezelfde namen, moet zoeken naar één bepaald bestand. Ook kan het zijn dat je een lijst wilt maken van bestanden in een bepaalde directory. Voor het maken van zo'n lijst gebruiken we de procedure FindNext. Deze procedure kan gebruikt worden nadat FindFirst aangeroepen is. Het volgende programma toont een lijst van bestanden op het scherm die gelezen worden uit de geldige (huidige) directory. De lijst vermeldt tevens de grootte van de bestanden, gevolgd door de datum en de tijd:

```
PROGRAM SCHIJF_2;
USES CRT, DOS;

VAR
   REC       : SearchRec;
   BESTAND   : PathStr;
   I         : Byte;
   DATUM_TIJD: DateTime;

   FUNCTION DatumStr(Dag,Maand,Jaar:Word):String;
   VAR
      CONVERT   : Word;
      STRCONVERT: String[4];
      DATUM     : String[10];
      I         : Byte;
   BEGIN
      DATUM := '';
      FOR I := 1 TO 3 DO
      BEGIN
         CASE I OF
            1: CONVERT := Dag;
            2: CONVERT := Maand;
            3: CONVERT := Jaar
         END;
         Str(CONVERT,STRCONVERT);
         IF Length(STRCONVERT) = 1 THEN
         STRCONVERT := '0' + STRCONVERT;
         DATUM := DATUM + STRCONVERT;
         IF I < 3 THEN DATUM := DATUM + '-'
      END;
      FOR I := Length(DATUM) TO 10 DO DATUM := DATUM + #32;
      DatumStr := DATUM
   END;

   FUNCTION TijdStr(Uur,Minuut,Seconde:Word):String;
   VAR
      CONVERT   :Word;
      STRCONVERT:String[4];
      TIJD      :String[8];
      I         :Byte;
   BEGIN
      TIJD := '';
      FOR I:= 1 TO 3 DO
      BEGIN
```

```
         CASE I OF
            1: CONVERT := Uur;
            2: CONVERT := Minuut;
            3: CONVERT := Seconde
         END;
         Str(CONVERT,STRCONVERT);
         IF Length(STRCONVERT) = 1 THEN
         STRCONVERT := '0' + STRCONVERT;
         TIJD := TIJD + STRCONVERT;
         IF I < 3 THEN TIJD := TIJD + ':'
      END;
      TijdStr := TIJD
   END;

BEGIN
   ClrScr;
   BESTAND := '*.*';
   FindFirst(BESTAND,0,REC );
   IF DOSError <> 0 THEN
   Writeln('Geen bestanden in directory.');
   WHILE DOSError = 0 DO
   BEGIN
      WITH REC, DATUM_TIJD DO
      BEGIN
         FOR I := Length(Name) TO 12 DO
         Name := Name + #32;
         UnpackTime(Time,DATUM_TIJD);
         Writeln(Name,'   ',Size:10,'  ',
         DatumStr(Day,Month,Year),'  ',
         TijdStr(Hour,Min,Sec))
      END;
      FindNext(REC );
   END
END.
```

Regels:Toelichting:

[1]3-7 Declareer de benodigde globale variabelen.
8-31Function DatumStr(Dag,Maand,Jaar:Word):String;
[8]14-30 Verander numerieke datumgegevens in een string.
32-54Function TijdStr(Uur,Minuut,Seconde:Word):String;
[8]38-53Verander numerieke tijdgegevens in een string.
55-73 Hoofdprogramma.
[2]57 Bestandsnaam is de wildcard voor alle bestanden.

```
   [3]58       Roep FindFirst aan.
59-60     Als DOSError ongelijk aan 0 nul is, zet dan een
          mededeling op het scherm.
   [4]61-72Als DOSError gelijk is aan 0, ga dan een WHILE-lus in
          die aangehouden wordt tot DOSError ongelijk is aan 0.
   [5]63  Maak door middel van het WITH-statement de velden van
          REC en van DATUM_TIJD rechtstreeks toegankelijk.
   [6]65       Maak het veld Name 12 bytes lang.
   [7]66       Vertaal het veld Time uit REC:SearchRec.
   [8]67-69Zet de bestandsnaam, de grootte, de datum en de tijd
          op het scherm.
   [9]71Roep FindNext aan voor het volgende bestand.
```

Toelichting:

[1]De globale variabelen die gedeclareerd worden, zijn bijna dezelfde als die uit het programma SCHIJF_1. Van de nieuwe variabelen wordt het veld I als index gebruikt, bestemd om dienst te doen in lussen. DATUM_TIJD is een variabele van het type DateTime. DateTime is een recordstructuur die in de unit DOS wordt gedefinieerd:

```
      DateTime = Record
   Year,Month,Day,Hour,Min,Sec: Word;
END;
```

Het type DateTime wordt gebruikt om het veld Time uit het type SearchRec te vertalen. Dit veld Time is een veld van het type Longint en is dus vier bytes lang. Door een inventief gebruik van de beschikbare bits, worden in dit ene veld zowel een datum als een tijdstip vastgelegd.

[2]In het hoofdprogramma beginnen we met het aangeven van de bestandsnaam. Om een lijst van alle bestanden in de directory te maken, moeten er jokers (wildcards) gebruikt worden. We kunnen hier de door het besturingssysteem gehanteerde jokers gebruiken. Een sterretje voor de punt betekent: alle namen. Het sterretje achter de punt betekent: alle extensies. Met *.* krijgen we dus alle bestanden uit de directory. Als we in bestandsnaam zouden zetten "C:*.*" , zoeken we naar alle bestanden in de root

directory van de C:-schijf.

[3]Als FindFirst aangeroepen is, kunnen we zien of er zich een bestand in de directory bevindt. Als er geen bestand gevonden wordt, heeft DOSError de waarde 18.

[4]Als er wel een bestand gevonden is, gaan we een WHILE-lus in. We blijven in deze lus tot DOSError een andere waarde dan 0 heeft. Als de waarde van DOSError ongelijk is aan 0, is er immers geen bestand meer gevonden.

[5]Met "WITH REC, DATUM_TIJD DO" maken we de velden van deze records rechtstreeks toegankelijk. Let op het plaatsen van de komma. Deze manier van werken met het WITH-statement werkt natuurlijk alleen als de velden in de verschillende records niet dezelfde naam hebben.

[6]Om de bestanden een beetje netjes op het scherm te krijgen, moeten de gegevens overzichtelijk in kolommen worden geplaatst. Daarom wordt er met strings gewerkt die voor iedere rij steeds even lang zijn. Het veld Name wordt op een lengte van 12 bytes gezet.

[7]Daarna wordt UnpackTime aangeroepen. UnpackTime krijgt als parameter het veld Time mee en als VAR-parameter het record DATUM_TIJD. UnpackTime is een procedure uit de unit DOS. Deze procedure haalt het veld Time uit elkaar en zet de vertaling in het record DATUM_TIJD. Met PackTime kun je het omgekeerde doen. Deze procedure maakt van de gegevens in het record een Longint.

[8]De velden in het record DATUM_TIJD zijn van het type Word. Dit betekent dat deze velden numeriek zijn. Om voor iedere rij een even lange string te krijgen, heb ik in dit programma de functies DatumStr en TijdStr geschreven. Deze functies retourneren een string die steeds even lang is voor de datum en de tijd. De functies worden als parameter meegegeven aan de Writeln-opdracht. Daarmee komt de geproduceerde string in de bedoelde vorm op het scherm.

[9]Nadat de bestandsgegevens op het scherm zijn gezet, wordt FindNext aangeroepen. FindNext krijgt als parameter het record REC mee. Als er geen bestand meer wordt gevonden, wordt DOSError op 18 gezet en wordt de lus verlaten. Indien DOSError de waarde 0 heeft, bevat REC de gegevens van een nieuw gevonden bestand.

9.4 Functies om de schijf te beheren

Naast de voorgaande methoden voor het zoeken van bestanden op schijf, biedt Turbo Pascal een groot aantal procedures en functies om vanuit je programma de schijf te beheren.

Het onderstaande programma SCHIJF_3 maakt een directory TEST aan als subdirectory van de root directory. Als je toevallig al een directory TEST hebt, geef deze directory dan een andere naam, of neem in het programma een andere naam op. Het programma plaatst in de directory TEST een bestand BESTAND.DAT. De naam van dit bestand veranderen we in BESTAND.TXT.

De variabele BESTAND wordt een volledig pad. Daarna wordt deze variabele gesplitst in een directory-pad, een naam en een extensie.

Daarna wordt BESTAND.TXT verborgen en laat het de werking zien van de attribuutparameter in FindFirst.

Tenslotte worden alle bestanden in de directory TEST verwijderd en uiteindelijk de directory zelf:

```
PROGRAM SCHIJF_3;
USES CRT, DOS;

VAR
   DIRNAAM, BESTANDSNAAM: PathStr;
   F                    : File;
   ZIN                  : String;
   ATTR                 : Word;
   DIR                  : DirStr;
   NAAM                 : NameStr;
   EXT                  : ExtStr;

PROCEDURE ToonDirectory(Attribuut:Word);
VAR
   REC: SearchRec;
BEGIN
   FindFirst('*.*',Attribuut,REC);
   IF DOSError = 0 THEN Writeln('Bestanden :')
   ELSE Writeln('Geen bestanden.');
   WHILE DOSError = 0 DO
   BEGIN
      Writeln(REC.Name);
      FindNext(REC)
   END;
   Writeln
END;


BEGIN
   ClrScr;
   Writeln
   ('De capaciteit van de hard disk = :',
     DiskSize(3),' bytes');
   Writeln
   ('Vrije schijfruimte  :',DiskFree(3),' bytes');
   DIRNAAM := 'C:\test ';
   BESTANDSNAAM := 'BESTAND.DAT';
   {$I-}
   MkDir(DIRNAAM);
   {$I+}
   IF IOResult = 0 THEN
      Writeln('Er is een directory ',DIRNAAM,' gemaakt')
   ELSE
      Writeln('De directory ',DIRNAAM,' bestaat al');
   ZIN := 'Dit is tekst voor in het bestand.';
   ChDir(DIRNAAM);
```

```
   Assign(F,BESTANDSNAAM);
   Rewrite(F,1);
   BlockWrite(F,ZIN,Length(ZIN));
   Close(F);
   Writeln(BESTANDSNAAM,' op schijf gezet');
   ToonDirectory(0);
   Reset(F);
   BESTANDSNAAM := 'BESTAND.TXT';
   Rename(F,BESTANDSNAAM);
   Writeln('Bestand is veranderd in ',BESTANDSNAAM);
   ToonDirectory(0);
   BESTANDSNAAM := FExpand(BESTANDSNAAM);
   Writeln
   ('Na aanroepen van FExpand staat in Bestandsnaam:',
     BESTANDSNAAM);
   Writeln;
   FSplit(BESTANDSNAAM,DIR,NAAM,EXT);
   Writeln('Na aanroep FSplit staat in Dir :',DIR,
           ' In NAAM staat :',NAAM,' In EXT : ',EXT);
   SetFAttr(F,Hidden);
   Writeln(BESTANDSNAAM,' is verborgen.');
   ToonDirectory(0);
   Writeln('Met het juiste attribuut naar FindFirst');
   ToonDirectory(Hidden);
   GetFAttr(F,ATTR);
   Writeln('De waarde van Hidden = ',ATTR);
   Erase(F);
   ToonDirectory(0);
   ChDir('\');
   RmDir(DIRNAAM);
   Readln
END.
```

Regels:Toelichting:

[1]3-10Declareer de benodigde globale variabelen.

11-24Procedure ToonDirectory(Attribuut:Word);

[7]14-23Laat de inhoud van de geldige directory zien en gebruik de parameter Attribuut als parameter voor FindFirst.

25-74Hoofdprogramma.

[2]27-29 Laat de capaciteit van de harde schijf zien.

[3]30-31 Laat de vrije ruimte van de harde schijf zien.

[4]32-33Geef de variabelen DIRNAAM en BESTANDSNAAM een waarde.

[5]34-40 Maak een directory en plaats een mededeling op het scherm.

[6]41-48Geef ZIN een waarde en plaats een bestand in de zojuist gemaakte directory. Zet de waarde die in de variabele ZIN staat in het bestand. Roep ToonDirectory aan om het resultaat te zien.

[8]49-53Geef het bestand een andere naam en laat opnieuw de directory zien.

[9]54-58Breid de waarde in de variabele BESTANDSNAAM uit tot het volledige pad en laat de nieuwe waarde van BESTANDSNAAM op het scherm zien.

[10]59-61Splits BESTANDSNAAM in een directory- , een naam- en een extensiegedeelte. Zet de verschillende waarden in de variabelen DIR, NAAM en EXT. Toon de waarden in deze variabelen op het scherm.

[11]62-64Verberg het bestand F en laat zien dat de procedure ToonDirectory nu meldt dat de directory leeg is.

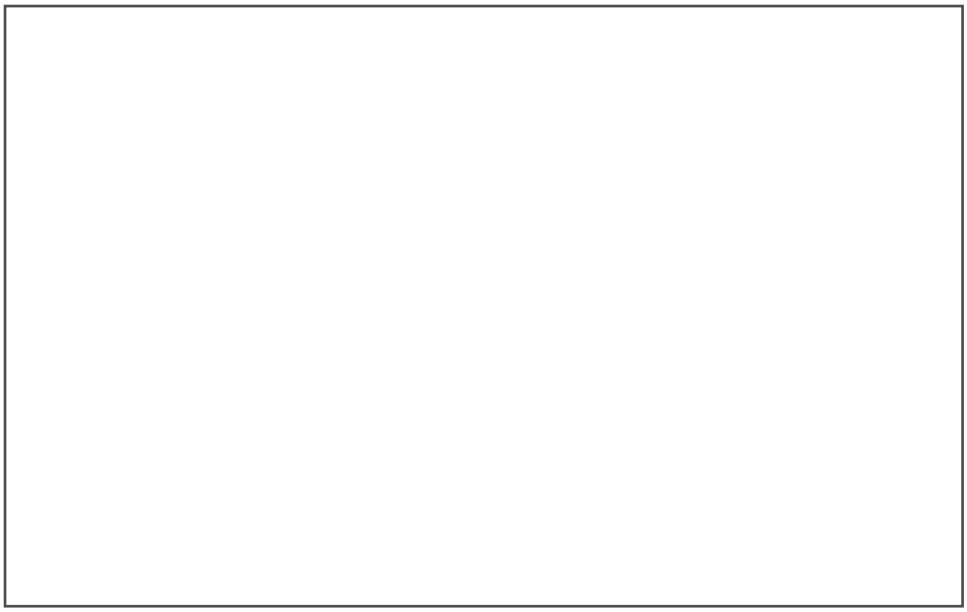
[12]65-66Roep nu ToonDirectory aan, met als parameter de waarde van de Turbo Pascal-constante Hidden.

[13]67-68 Haal de attribuutwaarde van het bestand op en laat dit op het scherm zien.

[14]69-70Verwijder het bestand uit de directory en laat vervolgens de inhoud van de directory zien.

[15]71 Maak de root directory de geldige directory.

[16]72 Verwijder de lege directory.



Afbeelding 12

Toelichting:

[1]De globale variabelen DIRNAAM en BESTANDSNAAM worden gebruikt om een directory te maken en te verwijderen, en om een bestand naar schijf te schrijven en te hernoemen. De variabele F is een ongetypeerde file. ZIN wordt als buffer gebruikt om met BlockWrite een tekst in het bestand te zetten. DIR, NAAM en EXT zijn bestemd om de verschillende delen van een pad in onder te brengen. DirStr, PathStr en ExtStr zijn in de unit DOS gedefinieerd als respectievelijk String[67], String[8] en String[4].

[2]Eerst wordt de functie DiskSize aangeroepen. DiskSize retourneert de capaciteit van de schijf in bytes. Als parameter krijgt DiskSize een nummer mee. Hierbij staat voor de A:-drive 1, voor de B:-drive 2 voor de C:-drive 3, enzovoort.

[3]De vrije ruimte op een schijf wordt berekend door de functie DiskFree. Ook deze functie krijgt als parameter een nummer mee dat de drive aanduidt.

[4]De directory die gemaakt wordt, wordt een subdirectory van de root directory en krijgt de naam TEST. Het bestand noemen we BESTAND.DAT.

[5]Met de optie {$I-} wordt de programma-onderbreking bij lees- en schrijffouten uitgeschakeld. De procedure MkDir uit de unit DOS probeert vervolgens op de aangegeven plaats een directory te maken. Na het aanroepen van MkDir wordt met {$I+}

het onderbreken weer uitgeschakeld. Als er al een directory TEST als subdirectory van de root directory bestaat, krijgt IOResult de waarde van de optredende DOS-fout. Als alles goed gaat heeft IOResult de waarde 0. Door het testen van de waarde in IOResult kunnen we nu zien of de directory reeds aanwezig was of dat de operatie gelukt is. Hiervan wordt melding gemaakt op het scherm.

[6]Nu we een directory hebben, kunnen we daar een bestand in zetten. De variabele ZIN krijgt een waarde. De bedoeling is dat deze tekst in het bestand gezet wordt. Vervolgens maken we door het aanroepen van de procedure ChDir de gemaakte directory tot de geldige directory. Ook ChDir is in de unit DOS gedeclareerd. Indien niets anders wordt aangegeven, worden gegevens gelezen en geschreven uit en naar de geldige directory. De Engelse term "current directory" wordt hiervoor veel gebruikt. We verbinden de variabele F met de waarde in BESTANDSNAAM en schrijven een nieuw bestand in de geldige directory. Door Rewrite een parameter mee te geven, zetten we de elementgrootte van het bestand op 1. BlockWrite zet vervolgens de inhoud van de variabele ZIN in het bestand. Op het scherm komt de mededeling dat het bestand op schijf is gezet. Vervolgens roepen we ToonDirectory aan om de inhoud van de directory op het scherm te plaatsen.

[7]In ToonDirectory worden met behulp van FindFirst en FindNext alle bestandsnamen in de directory naar het scherm geschreven. Is er geen bestand in de directory, dan wordt daar melding van gemaakt. Hier kun je duidelijk het voordeel van het werken met procedures en functies zien. In de loop van het programma wordt een aantal keer de inhoud van de directory getoond. Op deze manier kan dezelfde programmacode steeds opnieuw gebruikt worden.

[8]Teruggekeerd uit ToonDirectory wordt het bestand weer geopend en de inhoud van BESTANDSNAAM veranderd van BESTAND.DAT in BESTAND.TXT. De Turbo Pascal-procedure Rename verandert de naam van het bestand op de schijf. Rename krijgt als parameters de file die veranderd moet worden en de nieuwe naam mee. ToonDirectory wordt aangeroepen om te laten zien dat de naam van het bestand daadwerkelijk veranderd is.

[9]Soms is het handig te beschikken over de naam van het volledige pad. We hebben in de variabele BESTANDSNAAM alleen de waarde BESTAND.TXT gezet. Dit bestand is in de geldige directory geschreven. De geldige directory is "C:\TEST". Het volledige pad van ons bestand is dus "C:\TEST\BESTAND.TXT". De in de unit DOS gedeclareerde functie FExpand retourneert het volledige pad van de doorgegeven parameter. Het resultaat van de operatie tonen we op het scherm.

[10]Zoals het wel eens handig is om over de volledige naam van het pad te beschikken, zo is het ook wel eens handig om over een of meer fragmenten uit die naam te beschikken. Hiervoor wordt de procedure FSplit gebruikt. FSplit krijgt als eerste parameter het volledige pad door, gevolgd door drie VAR-parameters waarin achtereenvolgens het directory-pad, de naam van het bestand en de extensie gezet worden. Ook het resultaat van deze operatie wordt op het scherm getoond.

[11]Vervolgens willen we het bestand gaan verbergen. Hiertoe moet een bit omgezet worden in het attribuutveld van het bestandenrecord. Zoals we weten, houdt het besturingssysteem voor ieder bestand op schijf een dergelijk record bij. Het omzetten

van de bit wordt gedaan door de procedure SetFAttr. Door het doorgeven van de voorgedefinieerde constante Hidden, wordt het bestand op schijf verborgen. De procedure ToonDirectory geeft aan dat er geen bestanden in de directory aanwezig zijn. Als we de constante ReadOnly hadden doorgestuurd, dan kon het bestand alleen nog gelezen worden, en zou je er alleen in kunnen schrijven als de attribuutwaarde weer veranderd zou worden.

[12]Tot nu toe werd naar ToonDirectory steeds als parameter een 0 gestuurd. Sturen we echter de constante Hidden door, dan stuurt ToonDirectory deze paramater weer door aan FindFirst. Voor FindFirst is dit het teken dat naast de gewone bestanden ook de verborgen bestanden getoond moeten worden.

[13]De waarde van het attribuutveld van een bepaald bestand kan met behulp van GetFAttr uitgelezen worden. Hiertoe krijgt FAttr, naast de bedoelde file, ook een VAR-parameter mee om de gevonden waarde in te zetten. Uit de terugontvangen waarde blijkt dat de constante Hidden dezelfde waarde heeft als de door GetFAttr teruggegeven waarde.

[14]Nu we praktisch alle bewerkingen hebben gedaan, wordt het tijd om het bestand uit de directory te verwijderen. Dit karweitje wordt voor ons opgeknapt door de Turbo Pascal-procedure Erase. ToonDirectory laat zien dat de directory inderdaad leeg is.

[15]Om een directory te kunnen verwijderen, mag deze niet de geldige directory zijn. We moeten dus een andere directory geldig maken. De directory TEST is een subdirectory van de root directory. Als we nu van de root directory de geldige directory maken, kunnen we TEST verwijderen. Als aan ChDir de parameter "\" doorgegeven wordt, wordt de root directory, waarvoor de backslash het symbool is, de geldige directory. Als je aan ChDir twee puntjes ".." in plaats van een backslash doorgeeft, ga je één stapje hoger in de directory-hiërarchie.

[16]Nu de root directory de geldige directory is en de directory TEST leeg is, kunnen we de directory TEST gaan verwijderen. Dit klusje wordt uitgevoerd door de in de unit DOS gedefinieerde procedure RmDir.

9.5 Zoeken langs het pad

Soms weet je niet in welke directory een bepaald bestand staat. De functie FSearch uit de unit DOS brengt dan uitkomst. Het programma SCHIJF_4 zoekt langs een pad naar het bestand TURBO.EXE. Om het programma te testen, moet je de variabele PAD een waarde geven die overeenkomt met een zoekpad in je eigen computer:

```
PROGRAM SCHIJF_4;
USES CRT, DOS;

VAR
   PAD, GEVONDEN: PathStr;
   DIR: DirStr;

BEGIN
   ClrScr;
   PAD := 'C:\WP51;C:\BP\BIN';
   GEVONDEN := FSearch('TURBO.EXE',PAD);
   GetDir(0,DIR);
   Writeln
   ('TURBO.EXE staat in de directory:',GEVONDEN);
   Writeln('De geldige directory = ',DIR);
   Readln;
END.
```

Op de volgende bladzijde vind je de toelichting op dit programmaatje.

Regels: Toelichting:

3-5Declareer variabelen van het type PathStr om een zoekpad in op te geven en om een gevonden directory in op te slaan. Declareer een variabele DIR van het type DirStr voor gebruik met GetDir.

[1]8-9Zoek het bestand TURBO.EXE langs het opgegeven pad op.

[2]10 Geef de geldige directory door.

11-13Toon het gevonden pad en de geldige directory op het scherm.

Toelichting

[1]In de variabele PAD wordt, op de manier zoals dat in DOS gedaan wordt, een zoekpad opgegeven. De paden waarlangs gezocht moet worden, worden met puntkomma's van elkaar gescheiden. Samen met de naam van het gezochte bestand, wordt PAD naar de functie FSearch gestuurd. FSearch retourneert de directory waarin het gezochte bestand zich bevindt. De gevonden directory wordt in de variabele GEVONDEN gezet. Als het bestand niet gevonden wordt, retourneert FSearch een lege string.

[2]De procedure GetDir zet de ingestelde directory in de VAR-parameter DIR. Als eerste parameter krijgt GetDir het nummer van de drive mee waarvan we de ingestelde directory willen weten:

0 = Ingestelde drive
1 = A:-drive
2 = B:-drive
3 = C:-drive

9.6 Opgaven

9.1Maak een procedure waar je een te kopiëren bestand aan kunt doorgeven, en een directory waar het bestand in geschreven moet worden. Als het te kopiëren bestand niet gevonden wordt, moet daar melding van gemaakt worden en wordt de procedure

beëindigd. Als de directory voor de bestemming niet bestaat, dan moet gevraagd worden of de directory gemaakt moet worden. Als de vraag positief beantwoord wordt, moet het te kopiëren bestand gekopieerd worden naar de bedoelde directory. Als de vraag met nee beantwoord wordt, moet de procedure afgesloten worden.

9.2 Maak een programma dat het gekopieerde bestand en de aangemaakte directory van opgave 9.1 verwijdert.

161

161

161

161

182

182

182