24 Vermischtes

Fehlerbehandlung
Parameterübergabe
Zugriff auf die Umgebungsvariablen (Environment)
Aufruf eines anderen Programms mit Exec
Kommandozeilen-Compiler

Gliederung

24.1 Fehler und Fehlerbehandlung

In diesem Kapitel werden nur die Laufzeitfehler behandelt (das sind die Fehler, die beim Ablauf eines compilierten Programmes auftreten) und nicht die Fehler, die der Compiler während der Compilation entdeckt und danach die Compilierung abbricht.

Beim Auftreten eines Laufzeit-Fehlers wird das Programm mit der Meldung

`Runtime error` *`nnn`* `at` *`ssss`*`:`*`oooo`*

abgebrochen. *nnn* steht für einen maximal dreistelligen Fehlercode, *ssss* und *oooo* für die Segment- und die Offsetadresse (in hex) des Befehls, bei dem der Fehler auftrat. Diese Meldung ist dann von Nutzen, wenn man den Quellcode nicht besitzt und für die Fehlersuche ein eigenständiger Debugger, z.B. der Turbo-Debugger eingesetzt werden kann.

Turbo-Pascal unterscheidet zwei Klassen von Laufzeitfehlern:

- **Eingabefehler/Ausgabefehler** (Input/Output-Errors). Sie haben Fehlercodes zwischen 1 und 199 und können mit dem Compilerschalter `{$I-}` im Programm abgefangen und somit eventuell auch korrigiert werden.
- **Fehler mit sofortigem Abbruch** (Fatal Errors). Sie haben Fehlercodes zwischen 200 und 255 und sind durch programmtechnische Maßnahmen nicht korrigierbar.

24.1.1 Die Ein- und Ausgabefehler

Wenn man *vor* der Ein- oder Ausgabeoperation den Input/Output-Compilerschalter auf Minus setzt `{$I-}` (Standardeinstellung: `{$I+}`), dann können Ein- und Ausgabefehler mit der Standardfunktion »`IORresult`« abgefragt und unter Umständen behoben werden. Anschließend ist der Input/Output-Compilerbefehl wieder auf seine Standardeinstellung `{$I+}` zu setzen. In der Standardeinstellung wird bei jeder Input/-Output-Operation eine Prüfroutine aufgerufen, die im Fehlerfall das Programm mit dem zutreffenden Fehlercode abbricht.

Bei fehlerfreier Input/Output-Operation liefert `IOResult` den Wert 0, ansonsten einen Wert zwischen 1 und 199 nach der späteren Auflistung. Der Fehlercode kann auf eine Variable mit dem Datentyp Word zugewiesen werden. Letzeres ist notwendig, wenn nach mehr als einer Fehlermöglichkeit abgefragt werden muß, da die Funktion `IOResult` *nach* jedem Aufruf wieder den Wert 0 annimmt.

Dazu ein Beispiel:

```
program Pas24011;  { Druckerfehler }
uses
  CRT, PRINTER;
var
  Fehlercode: Word;
```

```
begin
  ClrScr;
  repeat
    {$I-}
    WriteLn(Lst, 'Ausgabeversuch auf Drucker');
    Fehlercode := IOResult;
    {$I+}
    case Fehlercode of
      0: WriteLn('K e i n  Druckerfehler');
    159: WriteLn('Druckerfehler 159: Out of Paper');
    160: WriteLn('Fehler 160: Allgemeiner Peripheriefehler');
    else WriteLn('Sonstiger Fehler, Fehlercode: ', Fehlercode);
    end;
    repeat until ReadKey <> '';
    if Fehlercode <> 0 then
       begin
         Write('Fehler beheben. Wenn fertig Taste Enter ....');
         repeat
         until ReadKey = #13;
             WriteLn; WriteLn;
       end;
  until Fehlercode = 0;
end.
```

Die Ein- und Ausgabefehler (genaue Beschreibung siehe Online-Hilfe):

2 File not found
3 Path not found
4 Too many files (zu viele Dateien gleichzeitig geöffnet. Siehe Config.SYS)
5 File access denied (Dateizugriff verweigert)
6 Invalid file handle (wahrscheinlich Dateivariable zerstört)
12 Invalid file access mode (ungültigerDateimodus)
15 Invalid drive number (unzulässige Laufwerkskennung)
16 Cannot remove current directory (Verzeichnis kann nicht gelöscht werden)
17 Cannot rename across drives (Rename nicht möglich)

100 Disk read error
101 Disk write error
102 File not assigned (Datei-Variable keinem DOS-Namen zugeordnet)
103 File not open
104 File not open for input
105 File not open for output
106 Invalid numeric format (ungültiges numerisches Format)

150 Disk is write protected (Diskette ist schreibgeschützt)
151 Unknown Unit (Peripheriegerät nicht bekannt/angeschlossen)
152 Drive not ready (Laufwerk nicht betriebsbereit)
153 Unknown command (ungültiger DOS-Code)
154 CRC error in data (Prüfsummenfehler beim Lesen Diskette/Platte)
155 Bad drive request structure length (ungültiger Disk-Parameterblock)
156 Disk seek error (Positionierfehler)
157 Unknown media type (unbekanntes Sektorformat)
158 Sector not found

159 Printer out of paper
160 Device write fault (Schreibfehler bei Peripheriegerät)
161 Devive read fault (Lesefehler bei Peripheriegerät)
162 Hardware failure (nicht genau bestimmbarer Hardware-Fehler)

24.1.2 Die Fehler mit sofortigem Abbruch

Fatal Errors. Genaue Beschreibung siehe Online-Hilfe

Diese Fehler lassen sich nicht korrigieren und haben Fehlercodes zwischen 200 und 255.

200 Division by zero (Division durch Null)
201 Range check error (Fehler bei der Bereichsüberprüfung)
202 Stack overflow error (Stackspeicher reicht nicht mehr aus)
203 Heap overflow error (Heapspeicher reicht nicht mehr aus)
204 Invalid pointer operation (fehlerhafte Zeiger-Operation)
205 Floating point overflow (Überlauf bei Fließkomma-Daten)
206 Floating point underflow (Unterlauf bei Fließkomma-Daten, bei 80x87)
207 Invalid floating point operation (Fließkomma-Fehler)
208 Overlay manager not installed (keine Overlay-Verwaltung)
209 Overlay file read error

24.2 Parameterübergabe

Jedem ausführbaren DOS-Programm (Extension *COM*, *EXE*, und *BAT*) können optional Parameter übergeben werden.

Zur Erinnerung: Das Formatieren einer Diskette im Laufwerk A kann z.B. wie folgt geschehen:

```
FORMAT  A:  /V:name
```

An den EXE-File »*FORMAT*« des Betriebssystem wurden zwei Parameter übergeben, einmal die Laufwerkskennung »*A:*« und der Switch »*/V*«, mit dem der Diskette nach dem Formatieren ein Name gegeben werden kann. Als Trennzeichen zwischen dem Befehl und den Parametern dienten ein oder mehrere Leerzeichen.

In ähnlicher Weise können an kompilierte Pascal-Programme (Dateien mit der Extension .EXE) beim Aufruf Parameter übergeben werden, auf die im Programm reagiert werden kann.

Für das Handhaben der Parameter dienen in Turbo-Pascal die beiden Funktionen

- `ParamCount`
- `ParamStr(`*`nummer`*`)`

Die Funktion »`ParamCount`« liefert die Anzahl der übergebenen Parameter mit dem Datentyp Word, wobei als Trennzeichen zwischen den Parametern nur zählen: Leerzeichen (eines oder mehrere) oder Tabulatoren.

Die Parameter werden immer als Strings übergeben. Gegebenenfalls sind Ziffernstrings im Programm mit »`Val`« in Numerik zu konvertieren. Die üblichen Trennzeichen für String-Konstanten (in Basic und C das Anführungszeichen »"«, in Pascal das Hochkomma »'«) werden nur als normale Textzeichen und nicht als Trennzeichen interpretiert.

Beispiel: Es liege ein Exe-File »`TEST.EXE`« vor. Beim Aufruf von der DOS-Kommandozeile aus mit:

```
TEST   Huber Maier
```

werden die *zwei* Parameter »`Huber`« und »`Maier`« übergeben. Die Funktion »`ParamCount`« liefert den Wert 2.

Beim Aufruf mit:

```
TEST  Huber    A:\Lager.DAT  '80335 München'
```

werden die *vier* Parameter »`Huber`«, »`A:\Lager.DAT`«, »'`80335`« und »`München'`« übergeben (das Hochkomma wird nicht als Trennzeichen interpretiert). Die Funktion »`ParamCount`« liefert somit den Wert 4.

Den Parameter-String selbst liefert die Funktion »`ParamStr(`*`nummer`*`)`«

Mit dem letzten Beispiel:

```
ParamStr(1) ———> Huber
ParamStr(2) ———> A:\Lager.DAT
ParamStr(3) ———> '80335
ParamStr(4) ———> München'
```

Wenn »*`nummer`*« (Word-Ausdruck) den Wert 0 hat, dann liefert diese Funktion den Namen des laufenden Programms.

Für die Programmerstellung in der Turbo-Pascal-Entwicklungsumgebung besteht die Möglichkeit, über Menüpunkt

»*Start/Parameter...*« im Fenster »*Kommandozeilenparameter*«

Parameter einzugeben und somit den Programmlauf des EXE-Files zu simulieren. Diese Einstellungen können jederzeit geändert werden. Sie sollten aber nicht mit »*Option/-Speichern*« abgespeichert werden.

```
program Pas24021; { Kap. 24.2: Parameterübergabe }
uses
  CRT;

var
  s:    string;
  i, n: Word;

begin
  ClrScr;

  n := ParamCount;

  if n = 0
     then WriteLn('Es wurde kein Parameter übergeben')
     else for i := 1 to n do
            begin
              WriteLn('Der Parameter ', i, ': ', ParamStr(i));
              if ParamStr(i) = 'Huber'
                 then WriteLn(#7, 'Parameter »Huber« übergeben');
            end;

  WriteLn('Der Parameterstring Nr. 0: ', ParamStr(0));
         { Der Name des laufenden Programms }

  repeat
  until ReadKey <> '';
end.
```

```
program Pas24022; { Parameterübergabe. Logarithmusberechnung }
      { Für praktische Anwendungen: Nach dem Kompilieren auf }
      { "Destination Disk"  EXE-File umbenennen in "Log.EXE" }
      { Aufrufbeispiel in DOS-Eingabezeile:   Log 0.3        }
uses
  CRT;

var   { In diesem einfachen Programm nur globale Variablen }
  n: Word;
  s: string;
  x: Real;
  Fehlercode: Word;

procedure Ausgabe;
begin
  WriteLn('Dek. Logarithmus von ', x, ':   ', Ln(x) / Ln(10));
  repeat
  until ReadKey <> '';
end;

procedure Eingabe;
begin
  repeat
    Write('Eingabe Zahl für Logarithmusberechnung > 0: ');
    ReadLn(s);
    Val(s, x, Fehlercode);
  until (Fehlercode = 0) and (x > 0);
end;

begin
  ClrScr;
```

```
  n := ParamCount;

  if n = 1
     then begin
            Val(ParamStr(1), x, Fehlercode);
            if (Fehlercode = 0) and (x > 0)
               then Ausgabe
               else begin
                      Eingabe;
                      Ausgabe;
                    end;
          end
     else begin
            Eingabe;
            Ausgabe;
          end;
end.
```

24.3 Zugriff auf die Umgebungsvariablen
Der Umgebungsspeicher (Environment)

Mit Hilfe der Unit DOS und der dort definierten Funktionen `EnvCount`, `EnvStr()` und `GetEnv()` kann in Turbo-Pascal auf die Variablen des Umgebungsspeichers (engl. Environment) zugegriffen werden. Das folgende Demo-Programm zeigt den Zugriff. Siehe auch Befehl SET im Kap. *Betriebssystem MS-DOS*.

```
program Pas24031;  { Environment-Test mit Unit DOS }
uses
  CRT, DOS;
const
  UmgebVariable = 'PATH';     { DOS: groß/klein beliebig }
  { 'COMSPEC', 'PATH', 'DIR', 'PROMPT', 'DIRCMD', 'TEMP' }
var
  i, iMax: Integer;
begin
  ClrScr;
  iMax := EnvCount; { Anzahl der Einträge in der Environmenttabelle }
  WriteLn('------ Das Environment ------- ');
  for i := 1 to iMax do   { Zählung ab 1 }
    WriteLn(i, '     ', EnvStr(i));

  WriteLn;
  WriteLn('Der Wert der Umgebungs-Variablen ', UmgebVariable,
          ': ', GetEnv(UmgebVariable));
  repeat
  until ReadKey <> '';
end.
```

24.4 Aufruf eines anderen Programms mit Exec

Mit der Standardprozedur Exec (Execute, benötigt Unit "DOS") kann innerhalb eines Pascal-Programms ein anderes ausgeführt werden. Beim Aufruf können optional Kommandozeilenparameter übergeben werden.

Format: `Exec(`*`programmname, parameter`*`)`

`programmname` Stringausdruck. Name des Programms. Zwingend mit Extension (.COM, .EXE, .BAT), ggf. mit Laufwerkskennzeichnung und Zugriffspfad.

`parameter` Stringausdruck. Kommandozeilenparameter. Wenn keine existieren, ist »´´« (leerer String) einzugeben, ansonsten müssen sie mit »**`/C`**« eingeleitet werden.

Fehler bei der Ausführung können über die Funktion `DOSError` (Unit DOS) abgefragt werden. Die mögliche Werte und die Erklärungen zeigt folgende Tabelle:

DOSError	Erklärung	DOSError	Erklärung
2	Datei nicht gefunden	8	Nicht genug Arbeitspeicher
3	Pfad nicht gefunden	10	Ungültige Umgebung
5	Zugriff verweigert	11	Ungültiges Format
6	Ungültiges Handle	12	Keine weiteren Dateien

Wichtig:

- Im Pascal-Programm, das `Exec()` benutzt, muß unbedingt eine maximale Heap-Einstellung vorgenommen werden, entweder über Menüeinstellung oder mit dem Compilerbefehl »`{$`**`M`** *`stack, heapMin, heapMax`*`}`«. Anderenfalls wird der gesamte Speicher als belegt angenommen und der Aufruf mit Exec führt zum `DOSError` 8: »Kein Platz im Hauptspeicher«. Die maximale Heapgröße sollte so klein wie möglich gewählt werden, ggf. auch 0, wenn das aufzurufende Programm großen Speicherbedarf besitzt.
- Turbo-Pascal besetzt einige Interrupt-Vektoren (Behandlung der Interrupt-Vektoren erst in Kap. "Systemnahe Programmierung"). Damit diese nicht vom aufgerufenen Programm benutzt oder verändert werden können, empfiehlt es sich, diese Interrupt-Vektoren vor dem Aufruf von `Exec()` mit den der Standardprozedur

  `SwapVectors`

  zu sichern und nach dem Aufruf mit der gleichen Prozedur wieder herzustellen. SwapVectors ist in der Unit DOS deklariert.

```
{$M 16384,0,0}     { Compilerbefehl $M wegen Begrenzung des Heaps
                     unbedingt notwendig. Hier "HeapMin" und
                     "HeapMax" auf 0, also kein Heap  }
program Pas24041; { Aufruf des MS-DOS-Kommandointerpreters }
                  { "Command.COM" mit und ohne Parameter   }
```

```
uses
  CRT, DOS;  { Unit DOS für: SwapVectors,
                             Exec(),
                             DOSError,
                             DOSExitCode }
procedure Fehlerbehandlung;
begin
  if DOSError <> 0
     then WriteLn(#7#13#10, 'DOSError: ', DOSError);
  if DOSExitCode <> 0 then
     begin
       WriteLn(#7#13#10, 'DOSExitcode: ', DOSExitCode);
       { Es folgt der Aufruf der Standardfunktionen "Lo()"  }
       { und "Hi()". Sie liefern als Byte-Typ den Low- bzw. }
       { High-Teil eines Word- oder Integertypen            }
       WriteLn('Der Low-Teil:  ', Lo(DOSExitCode));
       WriteLn('Der High-Teil: ', Hi(DOSExitCode));
       WriteLn('Der Low-Teil ist der "normale" Exitcode');
       WriteLn('Der High-Teil kann nur annehmen: ');
       WriteLn('  0 = Normales Programmende');
       WriteLn('  1 = Abbruch mit Strg-C   ');
       WriteLn('  2 = Gerätefehler         ');
       WriteLn('  3 = Keep-Prozedur für TSR');
     end;
  if (DOSError <> 0) or (DOSExitCode <> 0 )
     then repeat until ReadKey <> '';
end;

begin
  ClrScr;
  TextColor(Yellow);
  Write(#13#10#13#10, '1. Aufruf DOS-Kommandointerpreter ' +
                      'mit internem Befehl "dir": ');
  repeat until ReadKey <> '';
  WriteLn;

  SwapVectors;            { Interrupt-Vektoren sichern     }
  Exec('C:\Command.COM', '/C dir /W'); { Aufruf mit Befehl }
  SwapVectors;            { Wieder alte Interrupt-Vektoren }
  Fehlerbehandlung;

  Write(#13#10#13#10, '2. Aufruf DOS-Kommandointerpreter ' +
                      'ohne Befehl (DOS mit "exit" beenden) ... ');
  repeat until ReadKey <> '';

  SwapVectors;    { Interrupt-Vektoren sichern     }
  Exec('C:\Command.COM', ''); { Aufruf ohne Befehl }
  SwapVectors;    { Wieder alte Interrupt-Vektoren }
  Fehlerbehandlung;
end.
```

```
{$M 16384,0,0}  { Compilerbefehl $M wegen Begrenzung des Heaps
                  unbedingt notwendig. Hier "HeapMin" und
                  "HeapMax" auf 0, also kein Heap  }
program Pas24042; { Aufruf eines anderen Programms }
```

```
uses
  CRT, DOS;  { Unit DOS für: SwapVectors,
                             Exec(),
                             DOSError, DOSExitCode }
var
  Programmname,
  Parameter:    string;  { Kommandozeilen-Parameter }

procedure WriteXY(Sp, Ze: Byte; Meldung: string);
begin
  GotoXY(Sp, Ze);
  Write(Meldung);
end;

begin
  TextColor(Yellow); TextBackground(Blue);
  repeat
    ClrScr;
    WriteXY(5, 5, 'Demo: Aufruf eines beliebigen Programms ' +
                  'in einem Pascal-Programm');
    WriteXY(5, 6, 'Programmname komplett mit Zugriffspfad '  +
                  'und Extension (.COM, .EXE, .BAT)');
    WriteXY(5, 7, 'Beenden mit Enter-Taste alleine');

    WriteXY(5,  9, 'Programmname: '); ClrEoL;
    WriteXY(5, 10, 'Parameter:    '); ClrEoL;

    GotoXY(20,  9); ReadLn(Programmname);

    if Programmname = ''
       then Exit;      { >>>>>>>>>>>> }

    GotoXY(20, 10); ReadLn(Parameter);
    if Parameter = ''
       then Parameter := ''''
       else Parameter := '/C ' + Parameter;
    WriteLn;

    SwapVectors;  { Interrupt-Vektoren sichern }
    Exec(Programmname, Parameter);  { Aufruf }
    SwapVectors;  { Wieder alte Interrupt-Vektoren }

    WriteLn; WriteLn;
    if DOSError = 0
       then WriteLn('Hier Pascal-Programm nach dem Aufruf eines ' +
                    'anderen Programms ... ')
       else WriteLn('DOSError: ', DOSError);
    WriteLn('Der Exitcode des aufgerufenen Programms: ', DOSExitCode);

    repeat
    until ReadKey <> '';
  until Programmname = '';
end.
```

24.5 Der Kommandozeilen-Compiler

Bei großen Pascal-Programmen kann es vorkommen, daß der Arbeitsspeicher nicht mehr ausreicht, um das Programm zu compilieren. Besonders dann, wenn das Compilat auch im Arbeitsspeicher abgelegt wird. Abhilfe schafft in diesem Fall das Compilieren zu einer EXE-Datei. Aber auch dann kann es bei großen Programmen zu Problemen kommen, da die Integrierte Entwicklungsumgebung IDE (Integrated Development Environment) ebenfalls im Arbeitsspeicher steht. In diesen Fällen muß auf der Betriebssystem-Ebene der Kommandozeilen-Compiler benutzt werden. Die IDE kann zum Erstellen des Quellcodes benutzt werden; es kann dazu aber auch jeder beliebige Text-Editor benutzt werden, der ASCII-Dateien erzeugen kann.

Der Kommandozeilen-Compiler hat die Dateibezeichnung TPC.EXE und steht üblicherweise in BIN-Verzeichnis des Borland-Verzeichnisses BP.

Format: **`TPC`** [*`parameter/schalter`*] *`dateiname`* [*`parameter/schalter`*]

`tpc`	Kommandozeilen-Compiler TPC.EXE, ggf. mit Zugriffspfad
`parameter/schalter`	Müssen alle mit »/« eingeleitet werden und sind ansonsten fast vollständig mit denen der IDE identisch. Groß-/Kleinschreibung beliebig. Compilerbefehle und -schalter können aber auch im Quelltext vorhanden sein. Beim TPC-Aufruf sind die geschweiften Klammern wegzulassen.
`dateiname`	Name der Pascal-Quelldatei. Das Compilat bekommt den gleichen Namen, aber die Extension .EXE.

Beispiel: **`C:\BP\BIN\TPC`** `/$M 65520,0,0 C:\Student\Pas24031.PAS`

Der Stack wird in diesem Beispiel auf den Höchstwert 65520 gesetzt, HeapMin und HeapMax dagegen auf 0, also kein Heap.