

# READ WORK FILE

## Structured Mode-Syntax

```

READ WORK [FILE] work-file-number [ONCE]
    {
    RECORD operand1
    [AND] [SELECT]    { [ OFFSET n ]      operand2 }
                     { [ FILLER nX ] ...  } ... }
    [GIVING LENGTH operand3]
    [ AT [END] [OF] [FILE] ]
    [ statement ... ]
    [ END-ENDFILE ]
END-WORK
    
```

## Reporting Mode-Syntax

```

READ WORK [FILE] work-file-number [ONCE]
    {
    RECORD {operand1 [FILLER nX]} ...
    [AND] [SELECT]    { [ OFFSET n ]      operand2 }
                     { [ FILLER nX ] ...  } ... }
    [GIVING LENGTH operand3]
    [ AT [END] [OF] [FILE] ] { statement
                             [ DO statement ... DOEND ] }
[LOOP]
    
```

Dieses Kapitel behandelt folgende Themen:

- Funktion
- Feldlängen
- Verarbeitung dynamischer Variablen
- Syntax-Beschreibung
- Feldlängen
- Verarbeitung dynamischer Variablen
- Beispiel

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Verwandte Statements: CLOSE WORK FILE | DEFINE WORK FILE | WRITE WORK FILE

Gehört zur Funktionsgruppe: *Verarbeitung von Arbeitsdateien/PC-Dateien*

## Funktion

Das Statement READ WORK FILE dient dazu, Daten von einer physisch-sequentiellen Nicht-Adabas-Arbeitsdatei zu lesen. Die Daten werden sequentiell von der Arbeitsdatei gelesen. Wie sie gelesen werden, ist unabhängig davon, wie Sie auf die Arbeitsdatei geschrieben wurden.

Das READ WORK FILE-Statement führt eine Verarbeitungsschleife aus, um alle Datensätze der Arbeitsdatei zu lesen. Innerhalb einer READ WORK FILE-Schleife können automatische Gruppenwechsel-Verarbeitungen durchgeführt werden.

Informationen zu Unicode- und Codepage-Support siehe *Work Files and Print Files on Windows, UNIX and OpenVMS Platforms* in der *Unicode and Code Page Support*-Dokumentation.

### Anmerkungen:

1. Wenn bei Ausführung eines READ WORK FILE-Statements eine End-of-File-Bedingung auftritt, schließt Natural die Arbeitsdatei automatisch.
2. Bei Entire Connection: Beim Lesen von Entire Connection-Arbeitsdateien darf innerhalb der READ WORK FILE-Verarbeitungsschleife kein I/O-Statement stehen.

Wenn eine ASCII-Arbeitsdatei gelesen wird, kann es passieren, dass ein leerer Datensatz als letzter Datensatz nach dem letzten physischen Datensatz zurückgegeben wird. Dies geschieht, weil Natural keine individuellen Datensätze liest, sondern größere Blöcke der Arbeitsdatei, um die Performanz beim Datei-Zugriff zu optimieren.

## Feldlängen

Die Länge der Felder in der Operanden-Definitionstabelle wird wie folgt bestimmt:

Format	Länge
A, B, I, F	Die Anzahl der Bytes im Eingabedatensatz entspricht der internen Längendefinition.
N	Die Anzahl der Bytes im Eingabedatensatz ergibt sich aus der Summe der internen Stellen vor und nach dem Komma (Dezimalpunkt). Komma und Vorzeichen belegen im Eingabedatensatz kein Byte.
P, D, T	Die Anzahl der Bytes im Eingabedatensatz ergibt sich aus der Summe der Stellen vor und nach dem Komma (Dezimalpunkt) plus einer Stelle für das Vorzeichen, geteilt durch 2 und aufgerundet.
L	1 Byte wird benutzt. Bei Feldern des Formats C werden 2 Bytes benutzt.

**Beispiele für Feldlängen:**

Felddefinition	Eingabedatensatz
#FIELD1 (A10)	10 Bytes
#FIELD2 (B15)	15 Bytes
#FIELD3 (N1.3)	4 Bytes
#FIELD4 (N0.7)	7 Bytes
#FIELD5 (P1.2)	2 Bytes
#FIELD6 (P6.0)	4 Bytes

Siehe auch *Format und Länge von Benutzervariablen* im Leitfaden zur Programmierung.

## Verarbeitung dynamischer Variablen

### Syntax-Beschreibung

Operanden-Definitionstabelle:

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
<i>operand1</i>	S A G	A U N P I F B D T L C G	ja	ja
<i>operand2</i>	S A G	A U N P I F B D T L C	ja	ja
<i>operand3</i>	S	I	ja	ja

Bei dem Arbeitsdateityp ENTIRECONNECTION darf *operand2* nicht im Format C sein.

Siehe auch *Feldlängen*.

Syntax-Element-Beschreibung:

<b><i>work-file-number</i></b>	<p><b>Arbeitsdateinummer:</b></p> <p>Die für Natural definierte Nummer der Arbeitsdatei, die gelesen werden soll.</p> <p><b>Variabler Indexbereich:</b></p> <p>Wenn Sie ein Array von einer Arbeitsdatei lesen, können Sie für das Array einen variablen Indexbereich angeben. Zum Beispiel:</p> <pre>READ WORK FILE <i>work-file-number</i> #ARRAY (I:J)</pre>
--------------------------------	---

<p><b>ONCE</b></p>	<p><b>ONCE-Option:</b></p> <p>Die ONCE-Option bewirkt, dass nur ein Datensatz gelesen und keine Verarbeitungsschleife initiiert wird (das schleifenbeendende Schlüsselwort END-WORK bzw. LOOP darf daher nicht zusammen mit ONCE verwendet werden). Wenn Sie ONCE verwenden, sollten Sie auch eine AT END OF FILE-Klausel verwenden.</p> <p>Wird ein READ WORK FILE-Statement mit ONCE-Option von einer benutzerinitiierten Verarbeitungsschleife gesteuert, kann es sein, dass auf der Arbeitsdatei eine End-of-File-Bedingung auftritt, bevor die Schleife beendet wird. Alle von der Arbeitsdatei gelesenen Felder enthalten dann immer noch die Werte des zuletzt gelesenen Datensatzes. Die Arbeitsdatei wird dann zum ersten Datensatz zurückpositioniert, welcher dann bei der nächsten Ausführung des READ WORK FILE ONCE-Statements gelesen wird.</p>
<p><b>RECORD <i>operand1</i> FILLER <i>nX</i></b></p>	<p><b>RECORD-Option:</b></p> <p>Wird RECORD angegeben, so werden alle Felder jedes gelesenen Datensatzes zur Verarbeitung zur Verfügung gestellt. Entsprechend dem Aufbau des Datensatzes muss eine Operandenliste (<i>operand1</i>) angegeben werden.</p> <p>Mit FILLER <i>nX</i> geben Sie an, dass beim Eingabedatensatz <i>n</i> Bytes übersprungen werden sollen. Der in der RECORD-Klausel definierte Datensatz muss in einem zusammenhängenden Speicherbereich gespeichert sein.</p> <p>Im Structured Mode ist FILLER nicht erlaubt. Im Structured Mode — oder wenn <i>operand1</i> in einem DEFINE DATA-Statement definiert ist — darf <i>operand1</i> nur einmal angegeben werden. In beiden Fällen darf FILLER nicht verwendet werden.</p> <p>Natural überprüft die Daten des Datensatzes nicht. Demzufolge ist diese Option die schnellste Art, Datensätze von einer sequentiellen Datei zu verarbeiten. Allerdings müssen Sie selbst darauf achten, dass der Aufbau des Datensatzes korrekt beschrieben ist, da es sonst durch nicht-numerische Daten in numerischen Feldern zu einem ungewollten Programmabbruch kommen kann. Bevor der Datensatz gelesen wird, füllt Natural die Operandenliste (<i>operand1</i>) mit Leerzeichen; demzufolge wird bei einer End-of-File-Bedingung ein leerer Bereich zurückgegeben. Kurze Datensätze werden mit Leerzeichen aufgefüllt.</p> <p>Siehe <i>Übersicht für die Benutzung der RECORD-Option</i> weiter unten.</p>

<b>SELECT</b>	<p><b>SELECT-Option (Voreinstellung):</b></p> <p>Wird die <code>SELECT</code>-Option verwendet, so werden nur die in der Operandenliste (<i>operand2</i>) angegebenen Felder zur Verfügung gestellt. Die Position der Felder im Eingabedatensatz kann durch eine <code>OFFSET</code>- und/oder <code>FILLER</code>-Angabe spezifiziert werden.</p>			
	<table border="1"> <tr> <td data-bbox="605 384 829 457"><code>OFFSET n</code></td> <td data-bbox="837 384 1385 457"><code>OFFSET 0</code> spezifiziert das erste Byte des Datensatzes.</td> </tr> <tr> <td data-bbox="605 468 829 552"><code>FILLER nX</code></td> <td data-bbox="837 468 1385 552">Bedeutet, dass <i>n</i> Bytes des Eingabedatensatzes übersprungen werden.</td> </tr> </table>	<code>OFFSET n</code>	<code>OFFSET 0</code> spezifiziert das erste Byte des Datensatzes.	<code>FILLER nX</code>
<code>OFFSET n</code>	<code>OFFSET 0</code> spezifiziert das erste Byte des Datensatzes.			
<code>FILLER nX</code>	Bedeutet, dass <i>n</i> Bytes des Eingabedatensatzes übersprungen werden.			
<b>GIVING LENGTH <i>operand3</i></b>	<p>Natural ordnet den einzelnen Feldern die ausgewählten Werte zu und überprüft, dass die vom Datensatz ausgewählten numerischen Felder entsprechend ihrer Definition auch wirklich gültige numerische Werte enthalten. Aufgrund dieser Prüfung dauert die Verarbeitung einer sequentiellen Datei mit der <code>SELECT</code>-Option etwas länger als mit der <code>RECORD</code>-Option.</p> <p>Wenn ein Datensatz nicht alle in der <code>SELECT</code>-Option angegebenen Felder füllt, gilt folgendes:</p> <ul style="list-style-type: none"> <li>● Ein Feld, das nur teilweise gefüllt wurde, wird mit Leerzeichen bzw. Nullen aufgefüllt.</li> <li>● Ein Feld, das gar nicht gefüllt wurde, behält denselben Inhalt wie zuvor.</li> </ul> <p>Beim Lesen des Datei-Typs CSV werden die Optionen <code>OFFSET</code> und <code>FILLER</code> ignoriert.</p>			
	<p><b>GIVING LENGTH-Klausel</b></p> <p>Mit dieser Klausel können Sie die tatsächliche Länge des gelesenen Datensatzes in Erfahrung bringen. Die Länge (Anzahl der Bytes) erhalten Sie in <i>operand3</i>.</p> <p><i>operand3</i> muss mit Format/Länge I4 definiert sein.</p> <p>Wenn die Arbeitsdatei als <code>TYPE UNFORMATTED</code> definiert ist, verweist die zurückgegebene Länge auf die Anzahl der aus dem Byte-Strom gelesenen Bytes, einschließlich der beim <code>FILLER</code>-Operanden übersprungenen Bytes.</p> <p>Wenn die <code>GIVING LENGTH</code>-Klausel mit dem Arbeitsdateityp CSV benutzt wird, gibt der mit <code>GIVING LENGTH</code> angegebene Operand die Anzahl der Felder im Datensatz (und nicht die Länge des Datensatzes) an.</p>			

<b>AT END OF FILE</b>	<p><b>AT END OF FILE-Klausel</b></p> <p>Diese Klausel kann nur zusammen mit der ONCE-Option verwendet werden. Wenn Sie die ONCE-Option verwenden, dann sollten Sie auch eine AT END OF FILE-Klausel angeben, um festzulegen, was beim Auftreten einer End-of-File-Bedingung geschehen soll. Wenn Sie die ONCE-Option nicht verwenden, wird eine End-of-File-Bedingung wie das normale Ende einer Verarbeitungsschleife behandelt.</p>
<b>END-WORK</b>	<p>Das für Natural reservierte Wort END-WORK muss zum Beenden des READ WORK FILE-Statements benutzt werden.</p>

**Übersicht für die Benutzung der RECORD-Option**

<b>RECORD-Option wird benutzt mit</b>	<b>Zur Kompilierzeit zurückgewiesen</b>	<b>Zur Laufzeit zurückgewiesen</b>	<b>RECORD-Option wird ignoriert, Verarbeitung schaltet um in SELECT-Modus</b>
Arbeitsdateityp ENTIRE CONNECTION		x	
Dynamische Variablen	x		
Arbeitsdateityp CSV			x
Arbeitsdateityp PORTABLE			x
Arbeitsdateitypen ASCII, ASCII COMPRESSED, CSV, UNFORMATTED, Codepage ist im Configuration Utility angegeben (Konvertierung ist erforderlich) oder zumindest ein Unicode-Feld ist angegeben (Operand mit Format U, Konvertierung ist erforderlich)			x

**Feldlängen**

Die Länge der Felder in der Operanden-Definitionstabelle wird wie folgt bestimmt:

Format	Länge
A, B, I, F	Die Anzahl der Bytes im Eingabedatensatz entspricht der internen Längendefinition.
N	Die Anzahl der Bytes im Eingabedatensatz ergibt sich aus der Summe der internen Stellen vor und nach dem Komma (Dezimalpunkt). Komma und Vorzeichen belegen im Eingabedatensatz kein Byte.
P, D, T	Die Anzahl der Bytes im Eingabedatensatz ergibt sich aus der Summe der Stellen vor und nach dem Komma (Dezimalpunkt) plus einer Stelle für das Vorzeichen, geteilt durch 2 und aufgerundet.
L	1 Byte wird benutzt. Bei Feldern des Formats C werden 2 Bytes benutzt.

### Beispiele für Feldlängen:

Felddefinition	Eingabedatensatz
#FIELD1 (A10)	10 Bytes
#FIELD2 (B15)	15 Bytes
#FIELD3 (N1.3)	4 Bytes
#FIELD4 (N0.7)	7 Bytes
#FIELD5 (P1.2)	2 Bytes
#FIELD6 (P6.0)	4 Bytes

Siehe auch *Format und Länge von Benutzervariablen* im Leitfaden zur Programmierung.

## Verarbeitung dynamischer Variablen

Arbeitsdateityp	Verarbeitung
ASCII ASCII-COMPRESSED SAG(binär)	Die Arbeitsdateitypen ASCII, ASCII-COMPRESSED und SAG (binär) können keine dynamischen Variablen verarbeiten und rufen einen Fehler hervor. Sie können jedoch große Variablen mit einer maximalen Feld-/Datensatzlänge von 32766 Bytes verarbeiten.
ENTIRECONNECTION	Der Arbeitsdateityp ENTIRECONNECTION kann keine dynamische Variablen verarbeiten. Er kann jedoch große Variablen mit einer maximalen Feld-/Datensatzlänge von 107341824 Bytes verarbeiten.  Die RECORD-Option ist nicht erlaubt, wenn dynamische Variablen benutzt werden.
PORTABLE UNFORMATTED	Große und dynamische Variablen können mit den beiden Arbeitsdateitypen PORTABLE und UNFORMATTED in Arbeitsdateien geschrieben oder aus Arbeitsdateien gelesen werden. Bei diesen Typen gibt es keine Größenbeschränkung für dynamische Variablen. Große Variablen dürfen jedoch eine maximale Feld-/Datensatzlänge von 32766 Bytes nicht überschreiten.  Wenn eine dynamische Variable aus einer PORTABLE-Arbeitsdatei gelesen wird, so führt dies zu einer Größenänderung auf die gespeicherte Länge.
CSV	Die maximale Feld-/Datensatzlänge für dynamische und große Variablen ist 32766 Bytes. Dynamische Variables werden unterstützt. X-Arrays sind nicht erlaubt und resultieren in einer Fehlermeldung.

## Beispiel

```

** Example 'RWFEX1': READ WORK FILE
*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
*
1 #RECORD
  2 #PERS-ID (A8)
  2 #NAME (A20)
END-DEFINE
*
FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
  WRITE WORK FILE 1
    PERSONNEL-ID NAME
END-FIND
*
* ...
*
READ WORK FILE 1 RECORD #RECORD
  DISPLAY NOTITLE #PERS-ID #NAME
END-WORK
*
END

```



Ausgabe des Programms RWFEX1:

```
#PERS-ID      #NAME
-----
11100328 BERGHAUS
11100329 BARTHEL
11300313 AECKERLE
11300316 KANTE
11500304 KLUGE
11500308 DIETRICH
11500318 GASSNER
11500343 ROEHM
11600303 BERGER
11600320 BLAETTEL
11500336 JASPER
11100330 BUSH
11500328 EGGERT
```